VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Monitoring serverů pro firmu XO Studio

Servers monitoring for the company XO Studio

2012 **Libor Šusták**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Libor Šusták**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Monitoring serverů pro firmu XO Studio
Servers monitoring for the company XO Studio

Zásady pro vypracování:

Cílem práce je, seznámení studenta s tvorbou webových aplikací v jazyce PHP za pomoci YII frameworku, možnostmi monitorování serverů na základě různých typů protokolů (http, ftp, atd.). Vytvoření kompletní analýzy webové aplikace pro monitorování serverů, praktická realizace této aplikace a nasazení systému do reálného provozu.

Osnova:
1. Úvod do problematiky monitorování serverů
2. Bezpečnost komunikace.
3. Analýza systému:
   a) Specifikace požadavků.
   b) Datová analýza.
   c) Funkční analýza.
   d) Návrh implementace.
4. Praktická realizace systému.
5. Závěr.

Použité technologie: PHP, CSS, jQuery, Yii framework.

Seznam doporučené odborné literatury:

[1] GUTMANS, Andi; BAKKEN, Stig Sather; RETHANS, Derick. Mistrovství v PHP 5. Computer Press, 2007. 656 s. ISBN 978-80-251-1519-0, EAN: 9788025115190.
[2] CROFT, Jeff; LLOYD, Ian; RUBIN, Dan. Mistrovství v CSS : Pokročilé techniky pro webové designéry a vývojáře. Computer Press, 2007. 416 s. ISBN 978-80-251-1705-7, EAN: 9788025117057.
[3] HARWANI, B.M. JQuery Recipes : A Problem-Solution Approach. New York : Springer, 2010. 422 s. ISBN 978-1-4302-2709-0, ISBN-13 (electronic): 978-1-4302-2710-6.
[4] WINESETT, Jeffrey. Agile Web Application Development with Yii1.1. and PHP 5. 32 Lincoln Road Olton Birmingham, B27 6PA, UK. : Packt Publishing Ltd., 2010. 348 s. ISBN 978-1-847199-58-4.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Madecki**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka	prof. RNDr. Václav Snášel, CSc.
vedoucí katedry	*děkan fakulty*

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. 5. 2012 …….....

Poděkování

Na tomto místě bych velice rád poděkoval panu ing. Hrubému a ing. Bartošovi, za jejich velmi cenné a odborné rady. Dále bych chtěl poděkovat svým rodičům za podporu a to nejen při psaní bakalářské práce, ale za celé období mého studia. Nakonec patří velký dík i mé přítelkyni, která byla, je a doufám, že ještě dlouho bude mou velkou oporou.

Abstrakt

Hlavním cílem bakalářské práce je vytvoření webové aplikace pro monitoring serverů s využitím aplikační vrstvy firmy XO Studio. Aplikační vrstva firmy je postavena na open source frameworku Yii. Yii je objektově orientovaný PHP framework. Pro obsah a rozvržení stránek je použita technologie XHTML, pro vzhled prvků CSS a databáze MySQL. Aplikace umožňuje registrování uživatelů, následné přihlášení a přidávání URL adres s různými komunikačními protokoly. Tyto URL adresy jsou v daném časovém intervalu testovány a výsledky těchto testů jsou zaznamenávány do databáze. Registrovaný uživatel má poté k dispozici informace o těchto testech v podobě výpisů. Aplikační vrstva obsahuje administrační rozhraní, ve které je možno přidávat a měnit obsahové stránky, prvky menu a také spravovat záznamy z mnou implementovaných databázových tabulek.

Klíčová slova

PHP, MySQL, Yii, CSS, XHTML

Abstract

The main aim of this bachelor thesis is to create a web application for server monitoring using application layer of company XO Studio. Business application layer is built on open source framework Yii. Yii is an object-oriented PHP framework. For content and layout are used technologies XHTML, CSS for style elements and MySQL database. The application allows users register, login and subsequent addition of URLs with different communication protocols. These URLs are given time interval tested and the results of these tests are recorded in a database. Registered user then has available information about these tests in the form of statements. The application layer included administration interface, in which you can add and change content, menu elements and also manage the records implemented in database tables.

Keywords

PHP, MySQL, Yii, CSS, XHTML

Seznam použitých symbolů a zkratek

PHP	- Hypertext preprocessor
URL	- Uniform Resource Locator
MySQL	- Databázový systém
Yii	- Open-source PHP Framework
CSS	- Cascading Style Sheets
HTTP	- Hypertext Transfer Protocol
HTTPS	- Hypertext Transfer Protocol Secure
XHTML	- Extensible HyperText Markup Language
C	- Programovací jazyk
FI	- Form Interpretter
DOM	- Document Object Model
DTD	- Document Type Definition
GPL	- General Public License
OOP	- Object Oriented Programming
MVC	- Model-View-Controller
AR	- Active Record
SMTP	- Simple Mail Transfer Protocol
MTA	- Mail Transfer Agent
MUA	- Mail User Agent
POP3	- Post Office Protocol 3

Obsah

1. Úvod

Dokážete si v dnešní době představit svět bez internetu a bez všech komunikačních prostředků, které nám poskytuje? Nemálo z nás si určitě takový svět představit nedokáže a pro většinu lidí by takový svět nebyl světem v tom pravém slova smyslu. Poskytovatelé internetových služeb by tedy měli ve vlastním zájmu zajišťovat tyto služby co nejspolehlivěji a měli by proto své servery udržovat tak, aby byly stále dosažitelné. Zákazníci vlastnící webové stránky jistě nestojí o umístění na server, který je většinu času nedostupný.

Cílem bakalářské práce je vytvořit aplikaci pro monitoring serverů. Registrovaný uživatel bude mít v aplikaci možnost přidávat servery, u kterých chce sledovat dostupnost podle nastaveného intervalu měření a komunikačního protokolu. Aplikace v určitých časových intervalech zjišťuje, zda je zvolený server dostupný. Výsledné zjištění, ať už je pozitivní nebo negativní, je zaznamenáno do databáze. Uživatel poté získává v různých přehledech a statistikách veškeré potřebné informace získané sledováním. Informace mohou být užitečné nejen pro administrátory a správce sítí, ale také pro běžné uživatele.

Aplikace bude vytvořena pomocí open-source PHP frameworku Yii. Využita při tom bude aplikační vrstva poskytnutá firmou XO Studio.

V následujícím textu bude detailněji popsána celá aplikaci. Začnu pěkně od začátku a to tvorbou podrobné analýzy. Od slovního popisu a definování všech funkčních a nefunkčních požadavků se postupně propracuji k návrhu databáze. Navržená databáze přímo vybízí k vytvoření funkční analýzy, jejíž součástí budou Data Flow Diagramy. Následuje dynamická analýza a nakonec návrh implementace, kde je navržena kostra výsledné aplikace.

V další kapitole popisuji technologie použité při vývoji aplikace. Od známějších technologií jako jsou PHP, HTML nebo CSS se zaměřím na méně známou, ale pro tuto práci zásadní technologii, která nese honosný název Yii Framework. Nakonec v této kapitole nastíním pár detailů ohledně komunikačních protokolů, které jsou v rámci této bakalářské práce využívané.

S kompletně navrženou analýzou a popsanými technologiemi můžu přejít na samotnou implementaci aplikace. Zde se pokusím popsat aplikační vrstvu XO Studia a moduly, které jsem implementoval.

V závěru se podělím o své dojmy a pocity z mé práce. Zhodnotím, co jsem v aplikaci udělal, ale také doplním, čím by mohla být aplikace vylepšena.

Úplně posledním elementem práce jsou přílohy, kde se nachází všechny diagramy, obrázky a tabulky, které v analýze nenašly své místo.

2. Analýza

V analýze bude podrobně nastíněna strukturu celé aplikace pro monitoring serverů. Krok po kroku od slovního popisu a představ jak by měla aplikace vypadat, se analýza přehoupne k detailnějším rozborům jednotlivých procesů a funkcí. Nezbytnou součástí analýzy jsou i diagramy, které by měly nahradit a hlavně zpřehlednit kvanta textu. Při pohledu na diagramy by mělo být během okamžiku patrné, jak popisované funkce fungují.

2.1. Funkční požadavky

V této kapitole, jak už také její název vypovídá, jsou nadefinovány funkční požadavky systému. Jedná se o slovní popis a odpovědi na základní otázky při vývoji aplikace. Tyto základní otázky jsou proč, k čemu aplikaci potřebujeme a kdo s aplikací bude pracovat. Dále jsou určeny vstupy, kde budou popsány navrhované databázové tabulky. Jestliže jsou popsány vstupy, tak by neměly chybět ani výstupy a nakonec funkce systému, kde jsou uživatelé rozděleni do rolí a podle těchto rolí jsou definovány úkony, které každému uživateli náleží.

2.1.1. Proč potřebujeme tento systém?

Výsledná aplikace je využita především tehdy, je-li potřeba znát spolehlivost serverů, které jsou využívány. Důležité jsou zejména servery, za které platí naši zákazníci nebo lidé, pro které jsou tyto servery důležité. Proto je vyžadováno, aby byly servery efektivní, tedy spolehlivé a po většinu času plně funkční, dostupné.

Aplikace pro monitoring serverů v určitém časovém intervalu sleduje dostupnost zvoleného serveru.

2.1.2. K čemu systém potřebujeme?

Získáme kompletní a velmi užitečné informace o spolehlivosti provozu požadovaných serverů. Registrovaný uživatel tak může v případě nespokojenosti s určitým poskytovatelem služby, např. webhostingem, který má časté výpadky serveru ukončit spolupráci a přejít k jiné společnosti. Dále pak může využívat monitorovacích služeb této aplikace a vybrat si tak díky zprávám o dostupnosti ten nejspolehlivější server.

Díky statistikám o dostupnosti, získaných pomocí aplikace monitoring serverů tak uživatel může šetřit peníze a neplatit tak za nefunkčnost a nespolehlivost serveru.

2.1.3. Kdo bude se systémem pracovat?

Správce sítě, administrátor webových služeb, obyčejný uživatel. S aplikací pro monitoring serverů může pracovat v podstatě každý, do dané problematiky zainteresovaný, ale i nezainteresovaný uživatel.

2.1.4. Vstupy

Databázové tabulky, které budou v této kapitole popsány, jsou upraveny podle standardu firmy XO Studio pro návrh struktury a architektury databáze. Původní, ještě neupravený návrh databáze je možné s upraveným návrhem porovnat v kapitole popisující konceptuální model řešení.

Registrovaní uživatele jsou ukládáni do tabulky *tab_frontend_user*, kde každý uživatel má své ID, login a heslo, kterým se bude do systému přihlašovat, dále může vyplnit své jméno, příjmení a email, který slouží pro zasílání informací uživateli. Tabulka dále obsahuje atribut is_active, který nabývá hodnot 1 nebo 0. Tento atribut slouží pro viditelnost dat daného řádku a nastavuje se v administraci aplikace.

Tabulka *tab_server* slouží pro ukládání serverů přidaných uživatelem. O serveru budeme ukládat následující informace, ID serveru, cizí klíč z tabulky *tab_frontend_user* pro přiřazení serveru uživateli, který jej přidal, URL adresu serveru, který chceme sledovat, název, který slouží pro pojmenování serveru uživatelem, např. pro lepší orientaci ve vícero přidaných serverech a atribut is_active.

Tabulka *code_protocol* obsahuje pouze dva atributy. Tato tabulka slouží pro ukládání protokolů, které je možné v rámci aplikace pro monitoring serverů sledovat. Každý uložený protokol bude mít své ID, název protokolu a opět atribut is_active.

Tabulka *code_interval* je v počtu atributů stejná jako tabulka *code_protocol*. V tomto případě se ale jedná o tabulku, která uchovává záznamy o intervalech, ve kterých bude docházet k automatické kontrole dostupnosti serveru. Každý interval tedy má své ID, hodnotu intervalu a is_active.

Další, předposlední tabulkou, je tabulka *rel_server_interval_protocol*. Slouží k přiřazení protokolu a intervalu k uživatelem přidanému serveru. Tabulka uchovává ID, cizí klíč z tabulky *tab_server*, cizí klíč z tabulky *code_protocol* a cizí klíč z tabulky *code_interval*, posledním atributem tabulky je is_active.

Poslední tabulka nese název *tab_availability*. Tato tabulka slouží k ukládání záznamů z dotazovaných serverů, zda byly nebo nebyly dostupné a jaké návratové kódy byly při zjišťování dostupnosti obdrženy. V tabulce uchováváme následující informace, ID záznamu, tři cizí klíče na složený primární klíč z tabulky *rel_server_interval_protocol*, datum a čas pořízeného záznamu, odezvu, návratový kód a atribut is_active.

2.1.5. Výstupy

Výstupy budou v podobě výpisů a statistik, které bude moct uživatel řadit a filtrovat.
Např. přehled sledovaných serverů, přehled výpadků serverů, přehled dostupnosti serverů, statistiky dostupností a výpadků v daném období.

2.1.6. Funkce

V této podkapitole jsou uživatelé rozděleni do rolí, ve kterých se mohou během užívání aplikace pro monitoring serverů nacházet a jaké činnosti při tom mohou využívat.

Administrátor

- Odebrání uživatele
- Přidání uživatele
- Odebrání serveru
- Přidání serveru
- Přidání protokolu
- Odebrání protokolu
- Přidání intervalu měření
- Odebrání intervalu měření

Registrovaný uživatel

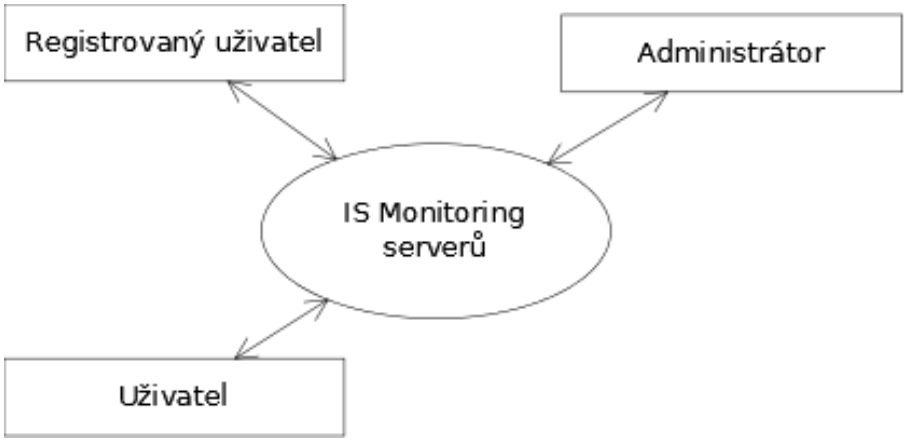
- Přidání serveru
- Odebrání serveru
- Zobrazení statistik a přehledů sledovaných serverů
- Nastavení protokolu
- Nastavení intervalu

Neregistrovaný uživatel

1. Registrace nového uživatele

2.1.7. Okolí

Se systémem budou pracovat, jak už naznačuje předchozí kapitola, běžní uživatelé, registrovaní uživatelé a administrátoři. Pro přehlednost a úplnost v této kapitole jistě nesmí chybět kontextový diagram, který znázorňuje vazbu mezi systémem a okolím. Nutno také podotknout, že kontextový diagram je základním stavebním kamenem Data Flow Diagramu, kterému se podrobněji věnuje část funkční analýza.



Obrázek 1 : Kontextový diagram

2.2. Nefunkční požadavky

U informačního systému se předpokládá přístup více uživatelů – klient server implementace, rychlá odezva. Přístup prostřednictvím internetu. Příjemné, jednoduché intuitivní ovládání a uživatelské rozhraní. Příjemný design.

Předem je stanoveno implementační prostředí Yii Framework využívající programovacího jazyka PHP. Dalšími požadovanými technologiemi jsou XHTML, CSS, MySql, jQuery.

2.3. Konceptuální model řešení

Nyní je potřeba data reálného světa formálně zapsat, navrhnout databázové tabulky resp. objekty, atributy a nakonec jednotlivé vazby mezi objekty.

Konceptuální model zahrnuje lineární zápis typů entit a jejich atributů, kde jsou zjištěny vztahy mezi entitami. Ještě přehledněji je možné vztahy mezi entitami vidět v ER diagramu a následně také v ER digramu s atributy. Jak již bylo zmíněno výše, všechny tabulky jsou upraveny podle specifikace firmy XO Studio. Do podkapitoly ER diagram s atributy je k porovnání zařazen i původní návrh databáze. Nakonec je popsán datový slovník.

2.3.1. Lineární zápis typů entit a jejich atributů

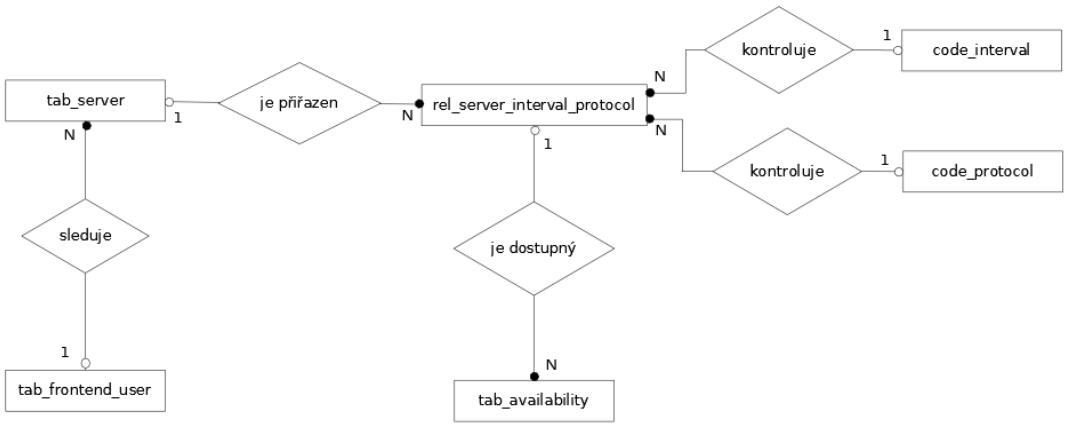
Každá databázová tabulka má, podle specifikace firmy XO Studio, svůj prefix podle toho o jaký druh tabulky se jedná. Každá tabulka pak musí obsahovat atribut **ID**. Jedná se o primární umělý klíč nabývající kladnou číselnou hodnotu. V některých případech se tento atribut může jevit jako zbytečný a redundantní, ale jak už bylo řečeno je vyžadován specifikací firmy, ale i aplikační vrstvou Yii frameworku.

Primární klíč je zvýrazněn **tučně**, cizí klíč *kurzivě*.

tab_frontend_user (**id**, login, password, name, surname, email, is_active)
tab_server (**id**, *id_tab_frontend_user*, url, title, is_active)
rel_server_interval_protocol (**id**, ***id_tab_server***, ***id_code_interval***, ***id_code_protocol***, is_active)
tab_availability (**id**, *id_tab_server_rel_server_interval_protocol*, *id_code_protocol_rel_server_interval_protocol*, *id_code_interval_rel_server_interval_protocol*, date_time, response, return_code, is_active)
code_protocol (**id**, protocol, is_active)
code_interval (**id**, interval, is_active)

Sleduje (tab_frontend_user, tab_server) 1:N
Je přiřazen (tab_server, rel_server_interval_protocol) 1:N
Je dostupný (rel_server_interval_protocol, tab_availability) 1:N
Kontroluje (rel_server_interval_protocol, code_protocol) N:1
Kontroluje (rel_server_interval_protocol, code_interval) N:1

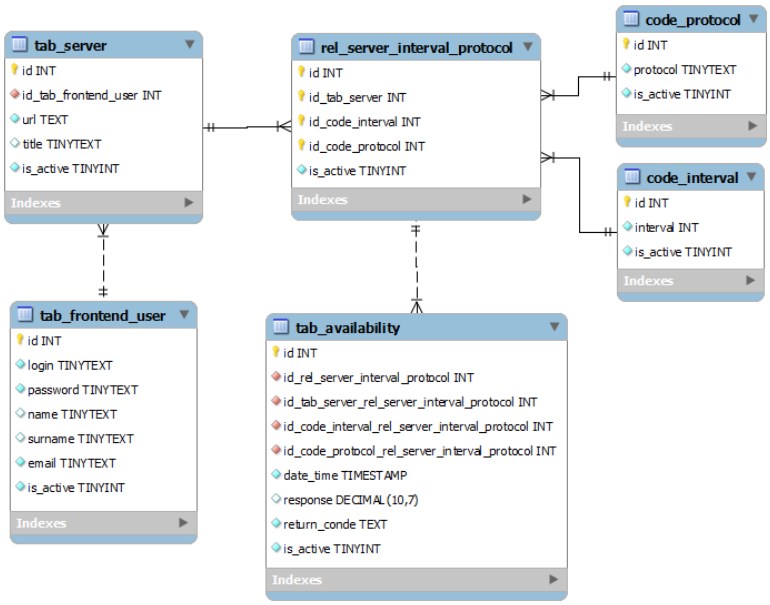
2.3.2. Úplný grafický tvar ERD



Obrázek 2 : Úplný grafický tvar ERD

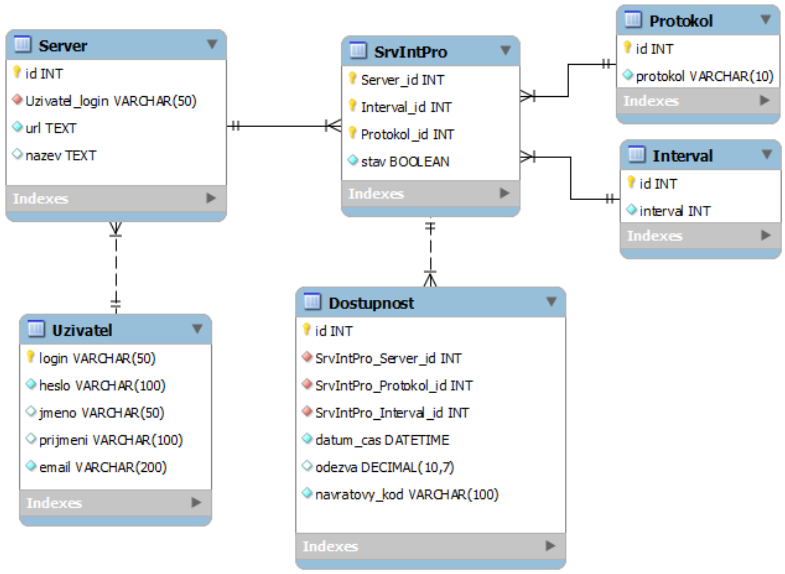
2.3.3. ERD diagramy s atributy

2.3.3.1. ER diagram podle specifikace firmy XO Studio



Obrázek 3 : ER diagram podle specifikace firmy XO Studio

2.3.3.2. ER diagram podle původního návrhu



Obrázek 4 : ER diagram podle původního návrhu

2.3.4. Datový slovník

tab_frontend_user:

Název	Datový typ	Klíč	Null	Popis
id	INT	PK	Ne	Jednoznačný identifikátor tabulky
login	TINYTEXT		Ne	Jednoznačný login uživatele
password	TINYTEXT		Ne	Heslo uživatele pro přihlášení
name	TINYTEXT		Ano	Jméno uživatele
surname	TINYTEXT		Ano	Příjmení uživatele
email	TINYTEXT		Ne	Email uživatele
is_active	TINYINT		Ne	Platnost záznamu

code_protocol:

Název	Datový typ	Klíč	Null	Popis
id	INT	PK	Ne	Jednoznačný identifikátor tabulky
protocol	TINYTEXT		Ne	Název protokolu
is_active	TINYINT		Ne	Platnost záznamu

code_interval:

	Datový typ	Klíč	Null	Popis
id	INT	PK	Ne	Jednoznačný identifikátor tabulky
interval	INT		Ne	Časový interval, ve kterém dochází k monitorování serveru
is_active	TINYINT		Ne	Platnost záznamu

tab_server:

Název	Datový typ	Klíč	Null	Popis
id	INT	PK	Ne	Jednoznačný identifikátor tabulky
id_tab_frontend_user	INT	FK	Ne	Identifikace uživatele, cizí klíč z tabulky tab_frontend_user
url	TEXT		Ne	Url adresa sledovaného serveru
title	TINYTEXT		Ano	Název serveru
is_active	TINYINT		Ne	Platnost záznamu

rel_server_interval_protocol:

Název	Datový typ	Klíč	Null	Popis
id	INT	PK	Ne	Jednoznačný identifikátor tabulky
id_tab_server	INT	PK,FK	Ne	Identifikace serveru, cizí klíč z tabulky tab_server
id_code_interval	INT	PK,FK	Ne	Identifikace intervalu, cizí klíč z tabulky code_interval
id_code_protocol	INT	PK,FK	Ne	Identifikace protokolu, cizí klíč z tabulky code_protocol
is_active	TINYINT		Ne	Platnost záznamu

tab_availability:

Název	Datový typ	Klíč	Null	Popis
id	INT	PK	Ne	Jednoznačná identifikace dostupnosti
id_tab_server_ rel_server_ interval_protocol	INT	FK	Ne	Identifikace serveru, cizí klíč z tabulky rel_server_interval_protocol
id_code_ protocol_rel_ server_interval_ protocol	INT	FK	Ne	Identifikace protokolu, cizí klíč z tabulky rel_server_interval_protocol
id_code_interval_ rel_server_ interval_protocol	INT	FK	Ne	Identifikace intervalu, cizí klíč z tabulky rel_server_interval_protocol
date_time	TIMESTAMP		Ne	Datum a čas
response	DECIMAL		Ano	Odezva
return_code	TEXT		Ne	Návratový kód
is_active	TINYINT		Ne	Platnost záznamu

2.4. Funkční analýza

Struktura databáze je již známa, nyní je čas podívat se na definování operací prováděných nad databází.

Následující řádky jsou zaměřeny na procesy a funkce navrhovaného systému. Jako první je popsán Data Flow Diagram nulté úrovně. Zde je pohled na jeden konkrétní proces, který je následně rozebrán do dalších úrovní DFD až po detailní minispecifikaci.

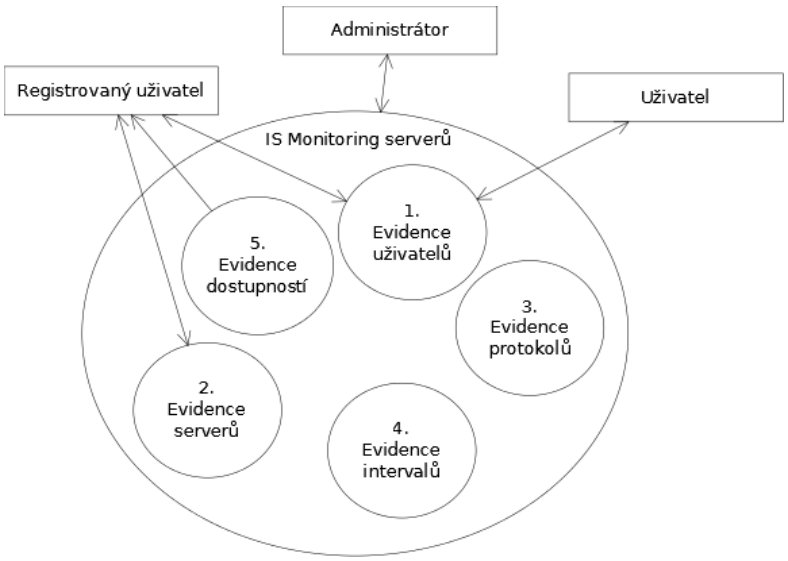
2.4.1. Data Flow Diagram

„Slouží jako grafický prostředek návrhu a zobrazení funkčního modelu systému – je základním nástrojem pro vyjádření konceptuálního funkčního modelu ve strukturovaných metodách. Funkční model popisuje, z jakých funkcí, jejich vstupů a výstupů se realita skládá a potažmo i jaké funkce budou tvořit informační systém, má-li být věrným modelem této reality.” [1]

Základní stavební kameny DFD jsou 4 komponenty: datová paměť, terminátor, datový tok a proces. Základem pro DFD je kontextový diagram, který již byl popsán v části funkční požadavky. Představíme-li si pyramidu, v přízemí naší pomyslné pyramidy se bude nacházet právě kontextový diagram. První patro obsadí DFD 0. úrovně. Tato úroveň je oproti kontextovému diagramu rozšířena o procesy vykonávané nad daným systémem. Pokud jsou procesy v systému dále dělitelné, přechází diagram do další úrovně a tím také stoupá vždy o patro výš v již zmiňované pomyslné pyramidě.

2.4.1.1. DFD 0. úrovně

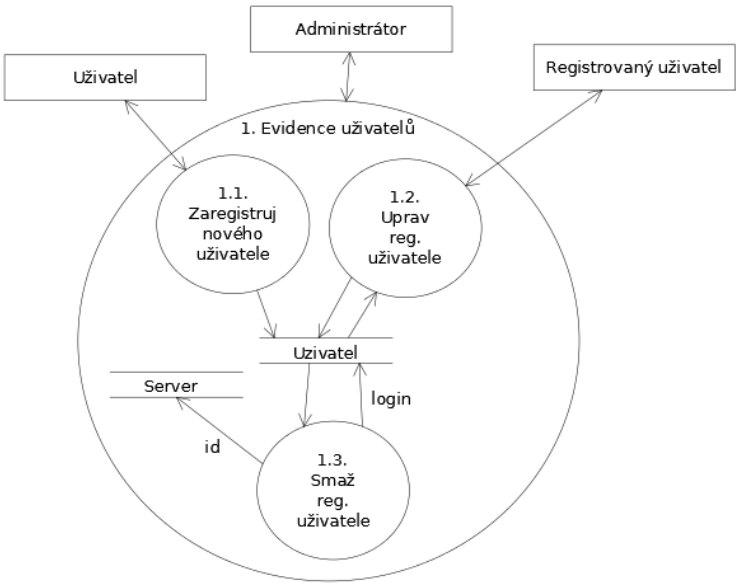
Na obrázku číslo 5, je znázorněn DFD 0. úrovně, který už je oproti již zmíněnému kontextovému diagramu popsán trošku rozsáhleji. Systém obsahuje 5 procesů a k nim vazby na jednotlivé terminátory. Všechny procesy jsou pro přehlednost očíslovány, což usnadní orientaci v dalších úrovních DFD.



Obrázek 5 : DFD 0. Úrovně

2.4.1.2. DFD 1. Úrovně – Evidence uživatelů

Další úrovní DFD je úroveň 1. Zde je pro ukázku vybran z 0. úrovně 1. proces. Původní proces Evidence uživatelů se skládá z dalších tří procesů, které jsou znázorněny na obrázku číslo 6.



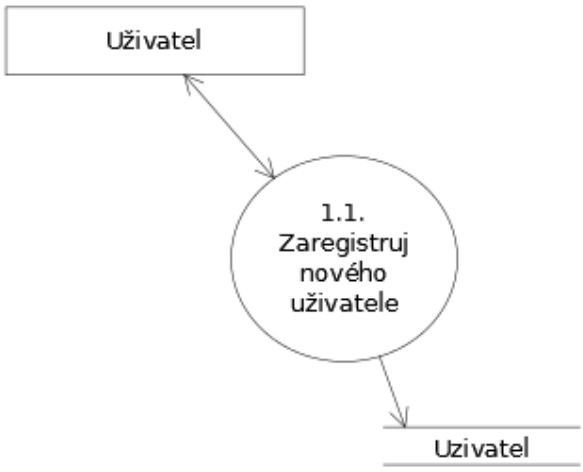
Obrázek 6 : DFD 1. Úrovně - Evidence uživatelů

2.4.1.3. Minispecifikace funkce 1.1. Zaregistruj nového uživatele

Od kontextového diagramu, přes nultou a první úroveň DFD je přechod k minispecifikaci. Pokud již nejde žádný proces DFD rozepsat do další úrovně, nastává okamžik, kdy jsou jednotlivé procesy popsány minispecifikací.

Je vybrán proces 1.1. Zaregistruj nového uživatele. Na obrázku číslo 7 je tento proces k vidění. Nachází se na něm terminátor Uživatel, spojen datovým tokem s procesem zaregistruj nového uživatele. Tento proces jak už jeho název napovídá slouží pro registraci nového uživatele. Pokud je proces vykonán, záznam je uložen do datového uložiště. Tento úkon je znázorněn datovým tokem směřujícím z procesu do komponenty datová paměť.

Pod obrázkem je popsán algoritmus vykonání procesu. Tento slovní popis algoritmu se nazývá minispecifikace. Minispecifikace popisuje to, jak je uživateli daný proces v reálném provozu reprezentován. Popisuje se krok po kroku interakce s uživatelem.



Obrázek 7 : Minispecifikace 1.1. – Zaregistruj nového uživatele

Algoritmus:

1. Zobraz formulář pro registraci uživatele

Login:	Uživatel zadá login
Heslo:	Uživatel zadá heslo
Jméno:	Uživatel může zadat jméno
Příjmení:	Uživatel může zadat příjmení
Email:	Uživatel zadá email

2. Uživatel vyplní formulář pro registraci
3. Zkontroluj údaje zadané uživatelem, vyskytne-li se chyba, jdi zpět na krok 2. Je-li všechno v pořádku, pokračuj dále
4. Vygeneruj celé kladné číslo pro atribut id
5. Zapiš vyplněný formulář jako záznam do tab_frontend_user
6. Konec cyklu pro jednoho uživatele

2.5. Dynamická analýza

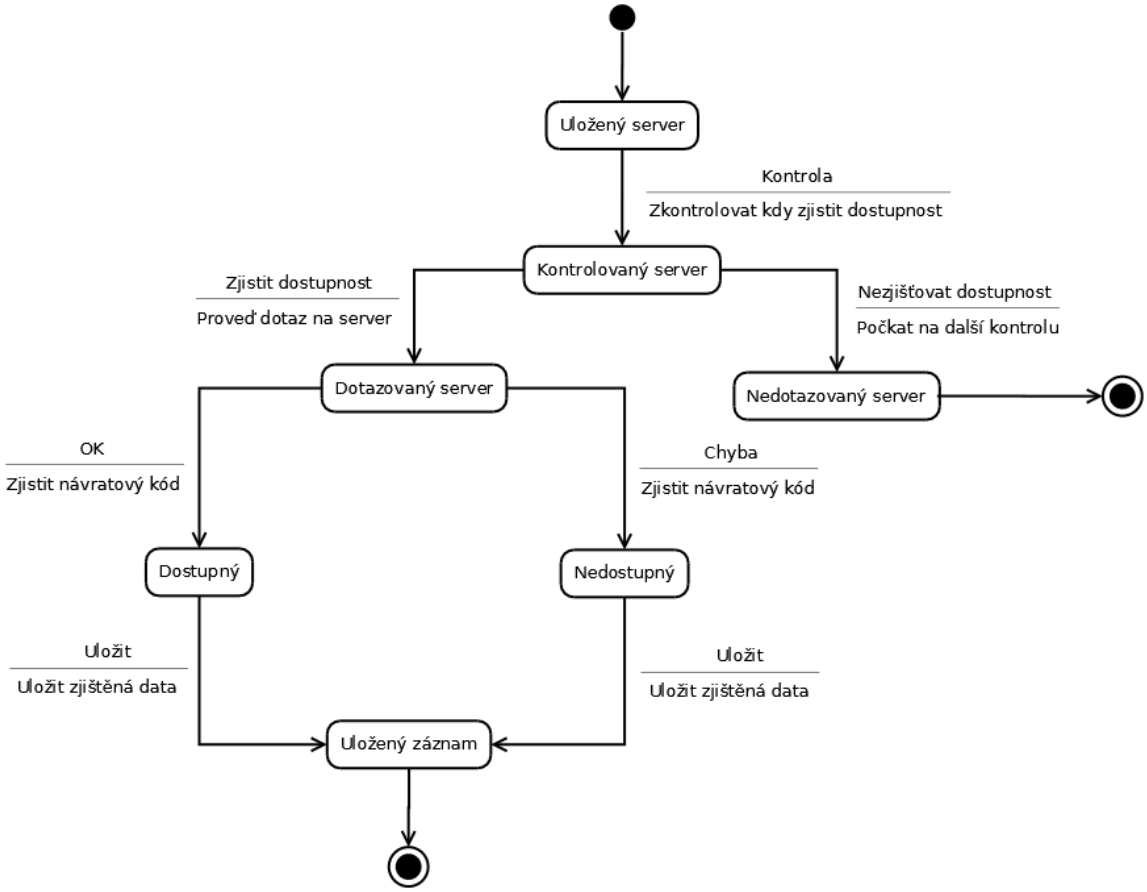
Doteď byly popisovány jen samotné funkce systému, ale funkce nemohou být spouštěny v libovolném pořadí, protože na sobě mohou být navzájem závislé. Dynamická analýza zachycuje stavy a chování systému v čase. Po vykonání určitého procesu, jak bylo popsáno v DFD, se systém nachází v nějakém stavu a právě tento stav je zachycen prostřednictvím dynamické analýzy. K nákresu je použit State Transition diagram(STD).

2.5.1. State Transition diagram

„Stavové diagramy jsou jednou ze známých technik pro znázornění chování systému. Popisují všechny možné stavy, které může nabývat konkrétní objekt systému, jinými slovy modelují chování objektu napříč všemi případy užití. Zároveň znázorňují, jak se stavy objektu mění v závislosti na událostech, které se ho dotýkají.

V mnoha OO technikách se stavové diagramy kreslí pro konkrétní třídu s cílem zobrazit životní cyklus konkrétního objektu. Existuje řada forem stavových diagramů, které se navzájem mírně odlišují sémantikou.” [2]

2.5.2. STD – dotaz na server



Obrázek 8 : STD diagram – dotaz na server

2.6. Návrh implementace

Požadavkem navrhované aplikace je běh v prostředí webového prohlížeče. Bude se jednat o webovou aplikaci využívající open-source PHP Framework Yii. Jak už bylo prozrazeno, základním tahounem Yii frameworku je skriptovací programovací jazyk PHP, dále HTML, CSS a jQuery. Pro ukládání dat je využit databázový systém MySql. Všechny zmíněné technologie budou blíže popsány v sekci použité technologie.

Nyní bude představeno přibližné grafické rozvržení jednotlivých komponent na stránce. Na počátku návrhu byla stanovena dvě základní zobrazení, která budou dále popsána.

2.6.1. Základní zobrazení

První dojem určitě hraje vždy velkou roli, a proto by základní zobrazení mělo na první pohled zaujmout a nalákat uživatele aby měl zájem stránku dále prozkoumávat. Toho docílíme příjemným a jednoduchým designem. Velmi důležité je také rozvržení komponent na stránce tak, aby se i nový uživatel během chvíle dokonale zorientoval.

Díky požadavkům na snadnou orientaci na stránce nebylo voleno ničím netypické rozvržení. Navrhovaná stránka se klasicky skládá z hlavičky, ve které je logo, uprostřed obrázek a napravo registrační a přihlašovací tlačítka. Pod hlavičkou se klasicky nachází obsahová část. Na levé straně se nachází menu, viditelné vždy pro všechny typy uživatelů. Vpravo obsahové části se nachází blok uživatelské tipy, kde si bude moct uživatel pomocí posuvníku zobrazovat rady a zajímavé tipy. Spodní část stránky je opět klasicky vyplněna patičkou, která obsahuje menu, copyright a další informace, které se v patičce často uvádějí.

Obrázek 9 : Návrh základního zobrazení

2.6.2. Uživatelské zobrazení

Uživatelské zobrazení se od zobrazení základního trošku liší. Uživatelské zobrazení se objeví až tehdy, přihlásí-li se uživatel do aplikace. Obrázek z hlavičky základního zobrazení zmizel. Je to tak lepší pro přehlednost výpisů, hlavička již nezabírá tolik místa. Tlačítka pro přihlášení a registraci nejsou potřeba, protože je uživatel přihlášen. Tyto dvě tlačítka jsou tedy nahrazena jedním s popiskem odhlásit. Na úrovni obsahu na pravé straně přibylo uživatelské menu, které umožňuje přidávat a měnit data.

Obrázek 10 : Návrh uživatelského zobrazení

3. Použité technologie

Základní technologií, která je v bakalářské práci využíta je objektově orientovaný open source PHP Yii framework, kterému bude v této kapitole věnována největší pozornost. Další použité technologie, které jsou využívány v rámci Yii jsou pochopitelně PHP, dále XHTML a CSS a jQuery. Tyto technologie budou samozřejmě také popsány. Zmíněn bude také databázový systému MySQL.

3.1. PHP

Tato kapitola je věnovaná velmi populárnímu, nejrozšířenějšímu skriptovacímu programovacímu jazyku pro tvorbu dynamických internetových stránek. Řeč je o PHP. Řádky níže odhalí něco z historie tohoto jazyka a stručně vysvětlí, oč se jedná.

3.1.1. Historie PHP

„U zrodu celého systému byl původně soukromý program Rasmuse Lerdorfa. Napsal si jednoduchý systém pro své vlastní webové stránky, především pro sledování návštěvnosti. Kvůli zvýšení výkonnosti jej později přeprogramoval v C a uvolnil k používání pro několik svých známých. Těm se velmi zalíbil a požadovali stále nové vlastnosti a chodili s novými připomínkami, čímž udrželi systém při životě a při vývoji. Při jeho uvolnění pro používání mu byl dán název Personal Home Page, PHP. Velké obliby se dočkal především ve své druhé verzi, obohacené především o velmi snadnou manipulaci s daty z formulářů – PHP/FI 2.0. FI v této zkratce znamenalo právě Form Interpretter. Od roku 1998 byla k dispozici verze 3.0, oproti předchozím podstatně zrychlená a obohacená o nové funkce, především podporu mnoha databázových systémů, objekty, cookies atd. Tou dobou už rozhodně nelze mluvit o PHP jako o nástroji pro domovské stránky uživatelů, PHP je nasazováno především na tisících velkých serverů poskytovatelů obsahu – zpravodajské servery, archivy softwaru atd. Proto se již vžilo označení PHP a původní význam této zkratky zapadl. Aktuálně se nejnovější verze PHP nachází ve verzi PHP 5.4.1. ” [3]

3.1.2. Co je PHP?

„PHP je hypertextový preprocesor, který na serveru interpretuje stránky HTML s vlastními příkazy před jejich odesláním ke klientovi (obvykle je jím webový prohlížeč). To znamená, že PHP umožňuje vkládat vlastní skripty (krátké úseky kódu, ale i celé programy) přímo do hypertextových stránek. To není nic neobvyklého – do hypertextových stránek lze vkládat také například kód v JavaScriptu. Existuje zde však několik podstatných rozdílů. Jednak je PHP interpretováno na server, zatímco JavaScript je jazyk interpretovaný teprve klientem. Oba způsoby se v některých ohledech velmi podstatně liší.” [3]

3.2. XHTML

3.2.1. Co je XHTML?

„Zkratka XHTML znamená eXtensible HyperText Markup Language, tj. „rozšiřitelný hypertextový značkovací jazyk”. Srozumitelnější formou by se dalo říci, že se jedná o jazyk, založený na odkazech na jiné stránky (hypertext), jenž se píše značkami, definovanými pro daný

jazyk (markup). Rozšiřitelný je proto, že vznikl rozšířením ze staršího jazyka HTML, a že je možné jej dále rozšiřovat o nové funkce.” [4]

3.2.2. Rozdíly HTML vs. XHTML

„V XHTML na rozdíl od HTML musí být všechny tagy ukončené a to včetně nepárových, jako jsou <meta>, <link>, <br>, <br> nebo <img>. Zápis může mít více podob. Buď použijeme klasické (a validní) <img></img> nebo zkrácené <img/> nebo mírně upravené <img />. První způsob se nedoporučuje používat, zasíláme-li XHTML dokument s typem text/html. Druhý způsob, bez mezery, se nedoporučuje používat kvůli postarším prohlížečům, které by v takovém případě mohly vynechat poslední atribut, je-li nějaký uvedený.

V XHTML na rozdíl od HTML musí být všechny tagy a jejich atributy zapsány malými písmeny, a to z toho důvodu, že jsou takto deklarované v odkazované DTD (Document Type Definition) a X(HT)ML je case sensitive, tedy záleží na velikosti písem. Pokud bychom si deklarovali vlastní DTD, můžeme směle používat i velká písmena.

Všechny hodnoty atributů musí být uzavřeny do uvozovek.

Dokument musí začínat XML deklarací. Její použití není povinné, pokud je dokument kódován v UTF-8 nebo pokud určujeme kódování vyšší protokolem (http například).

Pokud potřebujeme pracovat s rámy, můžeme deklarovat XHTML 1.0 Frameset a pro jednotlivé stránky XHTML 1.0 Transitional.

XHTML dokument bychom měli zasílat s jiným MIME typem než klasické HTML dokumenty. ” [4]

3.3. CSS

„Kaskádové styly (CSS, Cascading Style Sheets) vznikly na konci roku 1996. Přes relativně dlouhou existenci této technologie bylo její využití ve skutečném světě webového designu donedávna omezeno na správu písem a barev. Toto omezení bylo způsobeno její nekonzistentní podporou v prohlížečích. Protože ne všechny prohlížeče pracovaly s CSS stejně (pokud vůbec), bylo pro designery velmi obtížné skutečný potenciál kaskádových stylů využít. Místo toho zde byla jistota HTML.

Nyní je podpora kaskádových stylů mnohem lepší, takže vývojáři začali používat jejich mnohé cenné funkce a přecházet od HTML k návrhu vytvořeného jen pomocí CSS. ” [5]

3.4. jQuery

jQuery je open source JavaScriptová knihovna, která zjednodušuje interakci mezi HTML dokumentem, nebo přesněji Document Object Modelem (DOM) a JavaScriptem. [6]

3.5. MySQL

„MySQL je databázový systém vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems, dceřinou společností Oracle Corporation. Jeho hlavními autory jsou Michael „Monty” Widenius a David Axmark. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně

šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru." [7]

3.6. Yii framework

Tato kapitola je věnována Yii frameworku, nad kterým je vytvořena aplikace pro monitoring serverů. Od vzniku frameworku se přechází k popisu technologií a nastíněna je i základní charakteristika.

3.6.1. Vznik Yii

Yii je nápadem jeho zakladatele Qiang Xue, který začal vývoj tohoto open source frameworku 1. ledna roku 2008. Před tím, než začal Qiang pracovat na nové myšlence Yii frameworku, po mnoho let vyvíjel a spravoval jiný framework, PRADO. Léta zkušeností a uživatelských ohlasů v něm vyvolalo potřebu po mnohem jednodušším, rozšiřitelnějším a výkonnějším frameworku založeném na PHP5 splňující stále rostoucí potřeby vývojářů aplikací. První alfa verze Yii byla oficiálně uvolněna v říjnu roku 2008. Jeho extrémně působivé výkonnostní metriky v porovnání s jinými frameworky založenými na skriptovacím jazyku PHP okamžitě přitáhly velmi pozitivní pozornost. Dne 3. prosince roku 2008, bylo oficiálně představeno Yii 1.0. Zatím poslední stabilní verze je Yii 1.1.10, která byla uvolněna 12. února roku 2012. Připravuje se verze Yii 2.0, která je ovšem ještě ve stádiu vývoje.

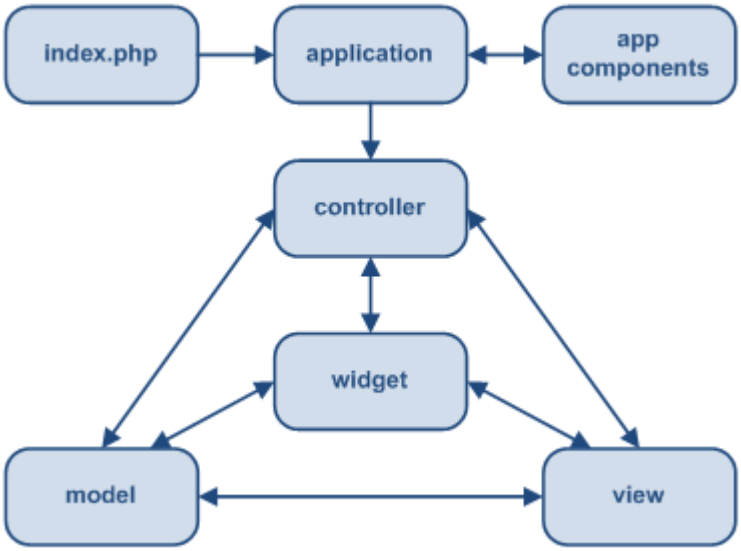
Pojmenování Yii pochází ze slovního spojení Yes, it is a vyslovujeme jej [ji:]. Yii má být také jakýmsi synonymem slov jednoduchý, efektivní a rozšiřitelný. [8]

3.6.2. Co je potřeba pro vývoj v Yii?

Pro běh aplikace založené na Yii jsou potřeba základní soubory frameworku, které jsou ke stažení na oficiálních stránkách Yii a webový server podporující PHP 5.1.0 a vyšší. Pro vytváření vlastních aplikací s Yii je potřeba znát pouze PHP a objektově orientované programování (OOP). Vytváření aplikací v Yii hlavně zahrnuje psaní a udržování svých vlastních PHP tříd, z nichž některé rozšiřují hlavní třídy Yii frameworku. [8]

3.6.3. MVC

Všechny aplikace vytvořené v Yii používají architekturu Model-View-Controller (MVC). To znamená, že aplikace je rozdělena, mimo jiné, do tří souborů, kterými jsou model, view a controller. V MVC, model představuje informace (data). View obsahuje prvky uživatelského rozhraní, jako je text, formuláře vstupů a controller řídí komunikaci mezi modelem a view. [9]



Obrázek 11: Struktura Yii aplikace [9]

Nyní bude blíže popsán obrázek 11. Ovšem ještě než se tak stane, musí být zmíněno, že všechny třídy v rámci Yii frameworku, začínají vždy písmenem C. To znamená, že defaultní aplikace v obrázku, je instancí třídy CWebApplication. Controller je zase instancí třídy CController nebo může být třída, která CController rozšiřuje. Tak zase zpět k obrázku. Na něm je k vidění struktura toho, jak aplikace pracuje s MVC architekturou. Nejprve je načten zaváděcí soubor index.php, který vytvoří instanci aplikace application. Základní aplikace Yii je tvořena adresářem protected. Aplikace v Yii jsou postaveny na komponentách. Pomocí komponent můžeme snadno spravovat funkcionalitu aplikace. Další krok vede na controller, který pomocí svých metod zachytává uživatelské požadavky. Zadá-li uživatel nějaký požadavek, je volána metoda controlleru action. Pokud není žádná akce specifikována, je volán defaultní controller, to znamená actionIndex. Controller poté může pracovat s modelem, který se stará o práci s daty. Nakonec je voláno view, které slouží pro zobrazování dat.

3.6.3.1. Model

Model je instance třídy CModel nebo jiná třída, která třídu CModel rozšiřuje. Modely se využívají pro práci s daty. Model představuje jeden datový objekt. Může být řádek v databázové tabulce nebo HTML formulář s uživatelskými vstupy. Každé pole objektu dat představuje atribut modelu. Atribut má označení a může být ověřen sadou pravidel rules.

Yii implementuje dva druhy modelů. Form modely a active records. Oba modely rozšiřují základní třídu CModel.

Form model je instance třídy CFormModel. Form modely se používají pro uchovávání dat získaných od uživatele prostřednictvím formulářových prvků.

Active Record (AR) je návrhový vzor používán pro abstraktní přístup do databáze v objektově orientovaném prostředí. Každý objekt AR je instance třídy CActiveRecord nebo podtřídy této třídy. Reprezentuje jeden řádek v databázové tabulce. [10]

3.6.3.2. View

View je PHP skript skládající se výhradně z elementů uživatelského rozhraní. Mohou obsahovat PHP příkazy, ale doporučuje se, aby tyto příkazy nijak nezasahovaly do datových modelů. Pro oddělení logiky a reprezentace dat by měly být operace nad modely umístěny v modelech nebo controllerech, nikoli ve view.

Název view se používá pro identifikaci souboru view při jeho renderingu. Název view je tedy shodný s názvem souboru. Například view s názvem edit bude odkazováno na soubor edit.php. Zobrazení view se dělá následujícím způsobem `CController::render()` s názvem požadovaného view.

Ve view můžeme volat instance controlleru použitím příkazu `$this->názevProperty`. [11]

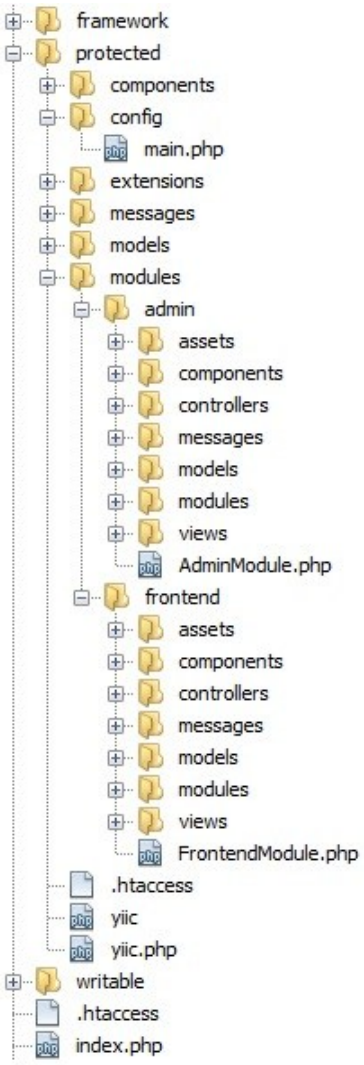
3.6.3.3. Controller

Controller je instancí třídy CController nebo jiné třídy, která rozšiřuje třídu CController. Je vytvořen objektem aplikace, když je na něj vznesen uživatelský požadavek. Controller většinou spolupracuje s modelem, od kterého získává data a ty poté renderuje pomocí view. Volaná akce je ve své nejjednodušší podobě metoda třídy controlleru, jejíž název začíná slovem action. [12]

3.6.4. Adresářová struktura

K úplnému pochopení Yii frameworku je zapotřebí porozumět i adresářové struktuře, která je celkem obsáhlá a tak pro nováčka může zprvu připomínat španělskou vesnici. Postupem času však určitě každý uzná, že tato struktura je navržena srozumitelně a přehledně.

První položkou adresářové struktury Yii je složka pojmenovaná framework. V této složce se nachází samotný framework a z pohledu vývoje aplikace pro monitoring serverů tato složka není nijak zajimavá. Mnohem důležitější je složka protected, která tvoří aplikaci. Nyní se popis zaměří právě na složku protected, konkrétně na to, co se v ní nachází. V components je možné nastavovat vlastnosti a volat a definovat události. Ve složce config se nachází soubor main.php, který je velice důležitý z pohledu nastavení aplikace. Tento soubor je zaveden při vytvoření aplikace, což je vždy, když se začne zpracovávat nějaký uživatelský požadavek. Soubor obsahuje název aplikace, importují se zde modely, componenty, moduly a další. Následuje popis modulů. Rozevřeny jsou dva moduly, admin a frontend. Každý tento modul obsahuje opět složku modules, ve které se nachází další moduly. Jak už je z názvů modulů admin a frontend patrné, tak modul admin obsahuje další moduly určené pro administrační rozhraní. Naopak modul frontend obsahuje moduly určené pro uživatele. Zaměřením se na modul frontend, je zjištěno, že se zde nachází složka assets, která obsahuje CSS, JavaScript a obrázky, které jsou využívány právě kaskádovými styly. Dále už se jedná o klasickou MVC architekturu. Controller obsahuje DefaultController.php, který slouží pro zobrazení úvodní stránky. Úvodní stránka se nachází ve složce view, kde jsou další složky, layouts a default. V layouts se nachází šablona stránek, logicky rozdělena na části, které definují základní dispozici webových stránek. Ve složce default se nachází index.php, který je určen pro zobrazení obsahu a je zasazen do již zmíněného layoutu.



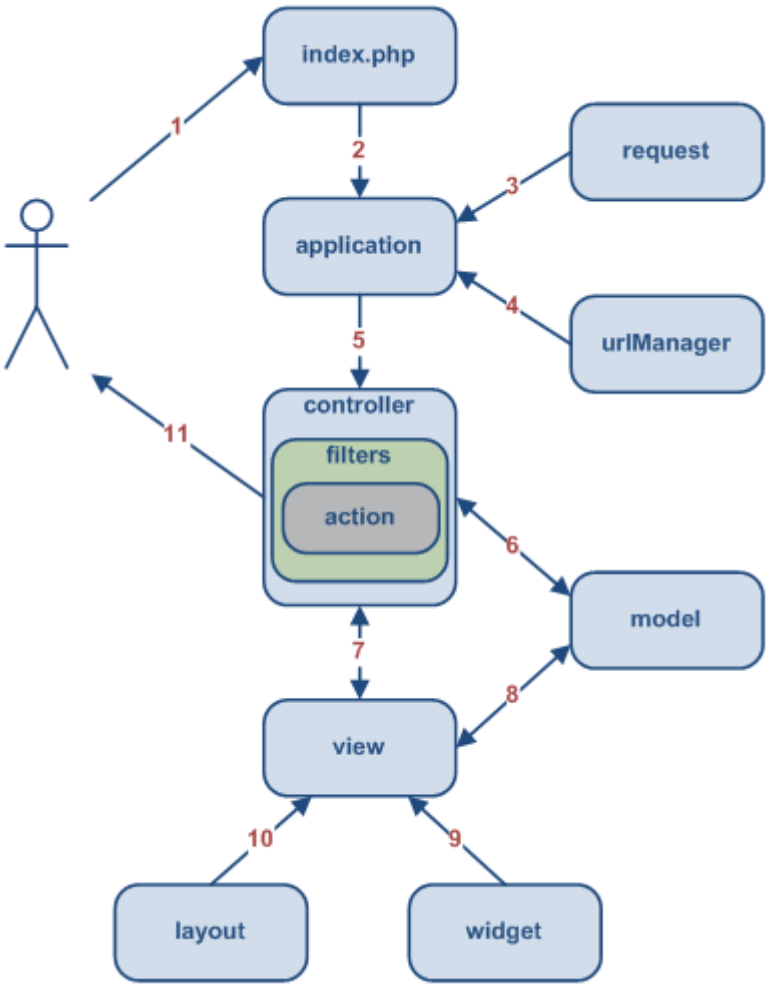
Obrázek 12 : Adresářová struktura Yii

3.6.5. Popis vykonání požadavku

Tato poslední kapitola věnovaná Yii frameworku se bude věnovat problematice zpracování požadavku uživatele. Čtenář bude seznámen s tím, co se děje v pozadí Yii frameworku, když uživatel zadá nějaký požadavek. Následující kroky patří k obrázku 13.

1. Uživatel vytvoří požadavek v podobě URL adresy `http://www.example.com/index.php?r=post/show&id=1`, webový server zpracuje požadavek provedením zaváděcího skriptu `index.php`.
2. Zaváděcí skript vytvoří instanci aplikace a spustí ji.

3. Aplikace získá podrobné informace o požadavku uživatele z aplikační komponenty zvané `request`.
4. Aplikace určí požadovaný `controller` a jeho `action` s pomocí aplikační component `urlManager`. V tomto příkladu je controller `post`, který odkazuje na třídu `PostController` a action `show`.
5. Aplikace vytvoří instanci požadovaného controlleru pro bližší zacházení s uživatelským požadavkem. Controller určí, že action `show` odkazuje na metodu jménem `actionShow` v controlleru.
6. Action načte z databáze `Post` model, kterému náleží ID 1.
7. Action vyrenderuje view `show` s `Post` modelem.
8. View přečte a zobrazí atributy `Post` modelu.
9. Pokud je ve view obsažen nějaký widget, zpracuje ho.
10. Vyrenderované view je vloženo do layoutu.
11. Action dokončí renderování view a zobrazí výsledek uživateli.



Obrázek 13 : Popis vykonání požadavku [9]

3.7. Komunikační protokoly

Tato část je zaměřena na komunikační protokoly, které je možné pomocí aplikace pro monitoring serverů testovat. Protokol je standard, který umožňuje přenos dat mezi dvěma koncovými body.

3.7.1. HTTP a HTTPS

„HTTP (Hyper Text Transfer Protocol) je internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML. Tento protokol je spolu s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech. HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků URL, který specifikuje jednoznačné umístění nějakého zdroje v Internetu. K protokolu http existuje také jeho bezpečnější verze označovaná jako HTTPS, která umožňuje přenášená data šifrovat a tím chránit před odposlechem či jiným narušením. Protokol funguje způsobem dotaz-odpověď. Uživatel (pomocí programu, obvykle internetového prohlížeče) pošle serveru dotaz ve formě čistého textu, obsahujícího označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server poté odpoví pomocí několika řádků textu popisujících výsledek dotazu (zda se dokument podařilo najít, jakého typu dokument je atd.) , za kterými následují data samotného požadovaného dokumentu.“ [13]

3.7.2. SMTP

„SMTP je zkratka anglického Simple Mail Transfer Protocol. To můžeme přeložit jako „jednoduchý protokol pro přenos pošty”. Jedná se skutečně o velmi jednoduchý protokol, který je ovšem pro odesílání a příjem veškeré elektronické pošty klíčový.

SMTP protokol používají pro komunikaci (odesílání pošty) jak klienti (MUA), tak i samotné servery (MTA) mezi sebou. Slouží vlastně k přenosu zpráv z jednoho počítače na druhý. Jeho syntaxe je velmi prostá, využívá k přenosu dat čistý text.“ [14]

3.7.3. POP3

„Post Office Protocol 3 (POP3 – poštovní protokol) je nejoblíbenějším klientským protokolem. Klient elektronické pošty jej používá pro příjem pošty na portu 110. Umožňuje uživatelům stahovat si svůj e-mail na lokální počítač. POP3 může rovněž umožnit lokální ukládání e-mailu na počítač uživatele i ukládání kopie zprávy na serveru samotném.

Server POP3 vyžaduje, aby měl každý uživatel své jméno a heslo a všechny zprávy většinou ukládá do jednotlivých textových souborů. Když je na tento server odeslána nový zpráva, kterou uživatel zatím neviděl, pak ji prostě doplní do uživatelova souboru. Server POP3 rozumí omezené sadě příkazů, která zahrnuje `user`, `pass`, `quit`, `list`, `retr`, `dele` a `top`.“ [15]

3.7.4. IMAP

„Internet Mail Access Protocol (IMAP – protokol přístupu k internetové poště) je naším zdaleka nejoblíbenějším protokolem, pokud se jedná o klientskou poštu. IMAP je další protokol používaný klientem pro připojení k serveru a převzetí e-mailu. IMAP používá port 143 a ukládá e-mail přímo na server.“ [15]

4. Implementace

Aplikace je zanalyzována, použité technologie už jsou také známy, teď přichází na řadu samotná implementace systému. Následující text je zaměřen na aplikační vrstvu firmy XO Studio. Také jsou popsány moduly vytvořené v administraci.

4.1. Popis aplikační vrstvy firmy XO Studio

Aplikační vrstva firmy XO Studio je velmi povedená a jak jsem se mohl sám přesvědčit, tak také velmi propracovaná. Asi největší chloubou celé aplikační vrstvy je, že při vývoji jsou pro databázové tabulky, které obsahují vícejazyčná data vytvářeny tabulky, které tyto vícejazyčná data uchovávají na základě cizího klíče. Taková vícejazyčná tabulka pak obsahuje ještě cizí klíč na tabulku jazyků. Nastane-li případ, že si zákazník firmy XO Studio usmyslí, že by chtěl obohatit své webové stránky o další jazyk, nenastává pro firmu v podstatě žádný problém. Aniž by totiž zákazník tušil, aplikace je vždy už od počátku vytvářena jako vícejazyčná a proto stačí pár malých úprav a je zde možnost pro volbu více jazyků.

Aplikační vrstva je založena na modulech. Největší výhodou na modulech je bezesporu jejich znovupoužitelnost a samostatnost daného modulu v rámci aplikace. Vezměme si za příklad modul komentáře. Jednoduše, je-li na stránkách potřeba používat komentáře, stačí opět jen pár úprav a začlenění modulu do aplikace je hotovo. Modul je totiž po funkční stránce hotový akorát je potřeba jej nainstalovat.

Součástí aplikační vrstvy je administrační rozhraní, kde se jednotlivé moduly spravují. Samozřejmě administrační rozhraní nabízí mnohem více funkcí, můžeme v něm, kromě vzhledu stránky, měnit v podstatě všechno od obsahu po prvky menu.

4.2. Moduly systému

V analýze, v části věnované DFD se nachází obrázek 5, který znázorňuje DFD 0. úrovně. Podle tohoto diagramu byly vytvořeny jednotlivé moduly. Procesy na obrázku 5 jsou rozloženy do dalších úrovní DFD a jsou definovány procesy pro přidání, odebrání a aktualizaci nad databází. Modul schovaný pod položkou menu v administračním rozhraní jakoby kopíroval již zmíněný proces v DFD 0. úrovně. Po kliknutí na položku menu se zobrazí data náležící danému modulu a jakoby posun do další úrovně DFD. V dalších úrovních DFD jsou procesy pro přidání, editaci nebo smazání záznamu a stejně tak i v administraci jsou k vidění tlačítka pro tyto funkce.

Moduly přidané do systému.

- **Uživatel** – obsahuje kompletní správu uživatelů, to zahrnuje přidávání, mazání a editaci. Tyto operace jsou prováděny nad tabulkou tab_frontend_user.
- **Protokol –** obsahuje kompletní správu seznamu protokolů, to zahrnuje přidávání, mazání a editaci. Tyto operace jsou prováděny nad tabulkou code_protocol.
- **Interval -** obsahuje kompletní správu seznamu intervalů, to zahrnuje přidávání, mazání a editaci. Tyto operace jsou prováděny nad tabulkou code_interval.
- **Server -** obsahuje kompletní správu serverů, to zahrnuje přidávání, mazání a editaci. Tyto operace jsou prováděny nad dvěma tabulkami tab_server a rel_server_interval_protocol. Tabulky jsou v modulu reprezentovány každý svým vlastním modelem. Sloučeny byly do jednoho modulu, protože je to tak z logického hlediska správné. Pro lepší pochopení je uveden například modul firmy XO Studio fotogalerie. Modul fotogalerie obsahuje modely fotogalerie a fotografie. Jelikož fotografie vždy náleží do nějaké fotogalerie, je logicky správné, že jsou umístěny

společně v jednom modulu. V aplikaci pro monitoring serverů jsou tabulky serverů, kde jsou uloženy jen servery. S protokolem a intervalem jsou servery vázány až tabulkou rel_server_interval_protocol.

- **Dostupnost** - obsahuje správu dostupností, to zahrnuje mazání a editaci. Tyto operace jsou prováděny nad tabulkou tab_availability. U tohoto modelu byla vyloučna možnost přidávání dostupností, protože záznamy o dostupnosti jsou přidávány automaticky a kdyby je někdo přidával ručně, tato funkčnost by ztrácela smysl.

4.3. Testování URL adres

Tato kapitola popisuje způsob řešení testování uživatelem zadaných URL adres. Aplikace by měla v určitých časových intervalech zkoumat dostupnost serverů. Pro tento účel byl vytvořen controller, který je v prohlížeči spouštěn aplikací operačního systému Windows Plánovač úloh (v linuxových operačních systémech se tento program nazývá Cron). Tento software umožňuje nastavit konkrétní čas nebo časový interval, kdy se má vykonávat určitá činnost. V tomto případě je nastaveno spouštění controlleru na opakování každých pět minut. Controller pak provádí testování URL adres.

4.3.1. Popis testovacího controlleru

Controller využívá návrhový vzor Factory. Návrhový vzor se skládá z rozhraní, které definuje metodu test s parametrem URL adresy viz obrázek 14.

```
interface ITesting {
  public function test( $url );
}
```

Obrázek 14 : Ukázka zdrojového kódu - ITesting

Pro každý testovaný komunikační protokol je definovaná konkrétní třída Tester. Třídy implementují rozhraní ITesting a tak každá obsahuje funkci test. Nyní bude popsán obrázek 15. Funkce test přijímá parametr v podobě URL adresy a k této adrese přiřazuje protokol HTTP, pro který je tato třída určená. Ještě před vykonáním dalších příkazů je funkcí microtime() zaznamenán čas v mikrosekundách. Na konci funkce test je tento čas zaznamenán ještě jednou, následně se tyto dva časy odečtou a výsledkem je čas, který je použit jako odezva serveru. Mezi již popsanými časy se nachází kód pro získání hlavičky HTTP protokolu. Následně je podmínkou zjištěno, zda byly obdrženy hlavičky nebo zda jsou prázdné. Pokud je proměnná headers prázdná return_code je nastaven na nedostupný. V opačném případě se zjišťuje kód vrácený v HTTP hlavičce. Pokud začíná číslem 2 nebo 3, je return_code nastaven na OK. To znamená, že server je dostupný. Čísla 2 a 3 jsou ořezané odpovědi obsažené v hlavičce a jedná se o třímístné číslo, které definuje stav serveru, v kterém se nachází. Třídy odpovědí 2XX (X jsou čísla, která konkretizují danou odpověď) znamenají kladnou odpověď a 3XX říkají, že stránka je přesměrovaná jinam, ale v tomto stavu je aplikací pro monitoring serverů považovaná za dostupnou. Nakonec funkce vrací pole obsahující odezvu serveru a návratovou zprávu.

```
class HTTPTester implements ITesting {
  public function test( $url ) {
    $fullUrl = "http://" . $url;
    $msTime  = microtime();
    @$headers = get_headers( $fullUrl );
    if( empty( $headers ) ) {
      $return_code = 'Nedostupný';
    } else {
      $return_code = substr($headers[0], 9, 3);
      if( $return_code[0] == '2' || $return_code[0] == '3' ){
        $return_code = "OK";
      } else {
        $return_code = "Nedostupný";
      }
    }

    $msTime2     = microtime();

    $response = round(($msTime2 - $msTime), 7);
    return array( 'return_code' => $return_code,
                  'response' => $response );
  }
}
```

Obrázek 15 : Ukázka zdrojového kódu - HTTPTester

Poslední součást návrhového vzoru Factory je třída TestFactory, která obsahuje funkci createTest s parametrem type. Tato funkce přijímá protokol a na základě tohoto protokolu je vybrán case, a vytvořen objekt třídy. Třída TestFactory je k vidění na obrázku 16.

```
class TestFactory {

  public function createTest( $type ) {
    switch( $type ) {
      case 'HTTP':
        return new HTTPTester;
        break;
      case 'SMTP':
        return new SMTPTester;
        break;
      default:
        return null;
    }
  }
}
```

Obrázek 16 : Ukázka zdrojového kódu - TestFactory

Návrhový vzor je poté využit v hlavní funkci controlleru actionIndex, která je spouštěna při spuštění plánovače úloh. V této metodě se prochází jednotlivé servery z databáze a zjišťuje se,

zda už někdy byly testovány podle dat z databázové tabulky tab_availability. Pokud server ještě nebyl testován, je proveden test, pokud už testován byl, zjistí se čas poslední kontroly, porovná se s intervalem měření, přiřazeném k serveru, a je-li čas k měření, tak je měření provedeno. Volání testovacích tříd s využitím návrhového vzoru Factory vypadá následovně.

```
$test = TestFactory::createTest($protocol);
$test->test($url);
```

Obrázek 17: Ukázka zdrojového kódu – Volání funkce test

4.3.2. Problém při testování URL adres

Při testování URL adres nastal problém, který se již nepodařilo opravit. Testování URL adres probíhá sekvenčně a to není zrovna dobré řešení. U serverů, které jsou dostupné, je výsledek testu znám v řádu milisekund. Problém nastává tehdy, když se testuje neexistující URL adresa. Podle provedených testů byl čas, než funkce zjistila, že server neexistuje v průměru 12 vteřin. Může se zdát, že 12 vteřin zase není až tak dlouhá doba, ale je-li v systému 1000 serverů, které mají být otestovány a třeba polovina z nich je nedostupná, tak čas pro vykonání skriptu by se musel několikanásobně zvýšit.

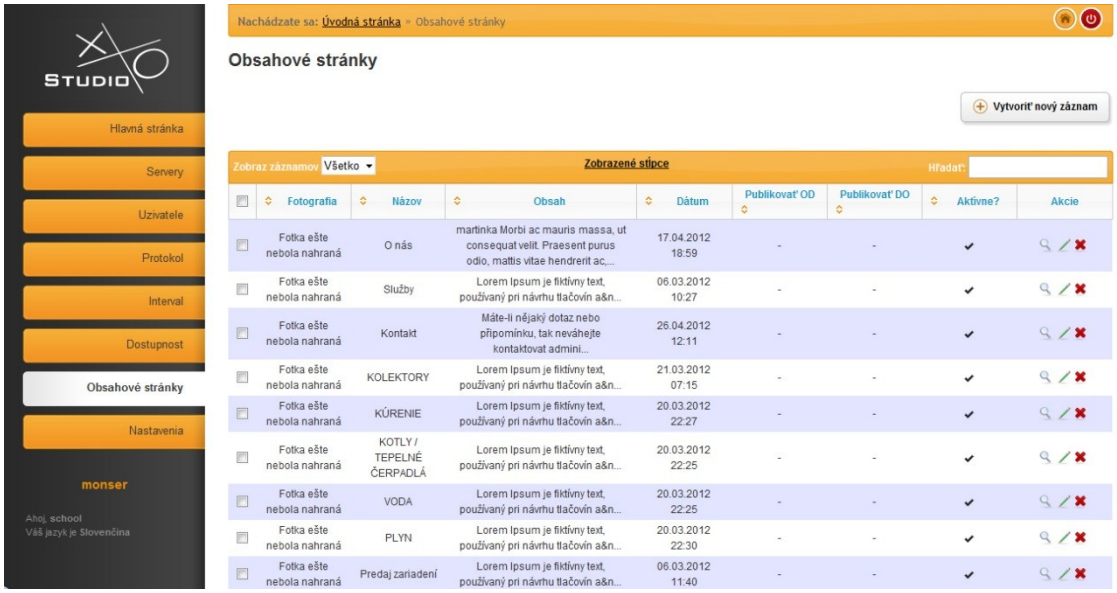
Nejdříve bylo zamýšleno, že bude pro každý test nastaven časový interval v řádech milisekund, ale některé funkce, které byly pro testy použity, tuto možnost nepodporovaly. Dalším řešením mohl být přechod na Unixový systém, který na rozdíl od systému Windows, na kterém byla aplikace vyvíjena, podporuje rozdělení kódu do nezávislých procesů pomocí funkce pcntl_fork. Zkoušena byla i možnost pingu na server, pomocí funkce exec, tahle možnost ale také nevedla k úspěchu a navíc je zde jisté bezpečnostní riziko.

Nejlepším řešením tohoto problému asi bude vytvoření nezávislé desktopové aplikace např. v programovacím jazyce JAVA nebo C#. Tato aplikace by běžela na serveru a starala by se čistě jen o testování serverů a následně by je ukládala do databáze.

Aplikace pro monitoring serverů tedy umožňuje alespoň sekvenční způsob testování. V konfiguračním souboru webového serveru byl alespoň změněn konfigurační soubor php.ini a maximální hodnota provádění skriptu, tedy max_execution_time byla zvýšena.

4.4. Ukázka administračního rozhraní

Zde je popsáno, jak vypadá administrační rozhraní a jak jsou zde zakomponovány moduly, které byly zmíněny výš. Na levé straně se nachází menu administračního rozhraní. Položky menu jsou jednotlivé moduly. Pokud klikneme na položku menu reprezentující modul, zobrazí se výpis modelu daného modulu. Záznamy poté můžeme přidávat, mazat a upravovat.

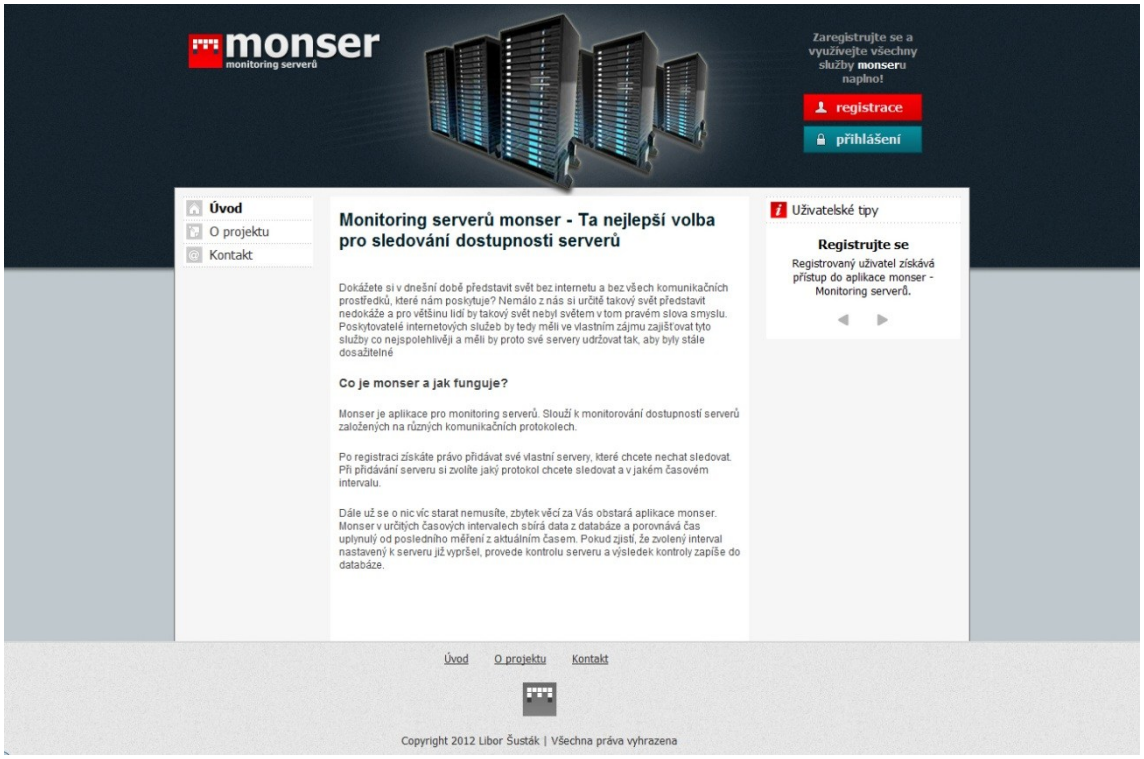


Obrázek 18 : Ukázka administračního rozhraní

4.5. Ukázka uživatelského rozhraní

Nesmí chybět ani ukázka uživatelského rozhraní neboli frontendu. Zatímco design administračního rozhraní je prací firmy XO Studio, uživatelské rozhraní je plně v mé režii. Snaha byla vytvořit design tak, aby byl pro uživatele co nejintuitivnější. V podstatě bylo dodrženo vše, co bylo definováno v návrhu implementace. Výsledné uživatelské rozhraní je vidět na obrázku číslo15.

Za zmínku určitě stojí i to, že menu na levé straně i menu dole v patičce jsou generovány z databáze a jejich text lze měnit v administraci, stejně tak jako obsah jednotlivých stránek.



Obrázek 19 : Ukázka uživatelského rozhraní

5. Zhodnocení

Ve skriptovacím programovacím jazyce PHP už jsem pár menších aplikací vytvořil, ale vždycky jsem měl nutkání své obzory rozšířit a naučit se programovat v PHP objektově a přinejlepším s využitím nějakého frameworku. Tuto šanci jsem dostal a rovnou ve formě bakalářské práce.

Nepředpokládal jsem, že bych se na Yii podíval a hned z fleku vytvořil dokonalou aplikaci. Možná jsem ale čekal, že programování a orientace v Yii bude ještě o něco snažší, než doopravdy je. Samozřejmě to byly a stále asi ještě jsou prvotní dojmy, protože si myslím, že jeden projekt ze mě specialistu neudělá.

Postupem času, i přes prvotní nezdary mě Yii framework vcelku zaujal a velký podíl na tom měl i ing. Hrubý, který dokáže věci, kterým opravdu rozumí s nadšením a srozumitelnou formou předat. Na Yii frameworku a také na aplikační vrstvě firmy XO Studio mě asi nejvíce zaujala možnost rychlého nasazení aplikace v případě, že už máte z dřívějších projektů vytvořené moduly. Potřebujete na stránkách komentáře, fotogalerii, odesílání newsletterů? To není žádný problém, moduly se jen nainstalují, doladí se pár detailů a stránky jsou hotové. Díky těmto možnostem a díky tomu, že Yii framework je zcela zdarma, tak nabízí relativně dobrou možnost výdělku. Ke škodě určitě není ani MVC architektura, která mi také dala z počátku trošku zabrat a než jsem se zorientoval, co vlastně dělá controller a co model, tak to chvíli trvalo.

Teď si akorát pokládám otázku, jak moc vlastně aplikační vrstva firmy XO Studio ovlivnila ten původní, čistý Yii framework. V budoucnu bych chtěl určitě zkusit vytvořit vlastní aplikaci postavenou na holém a nikým neupravovaném Yii.

6. Závěr

Cílem bakalářské práce bylo vytvořit webovou aplikaci pro monitoring serverů. Tento úkol jsem splnil, samozřejmě je zde dost nedostatků a věcí, které by se daly zlepšit popřípadě řešit jinak. Každopádně práce pro mě byla obrovským přínosem. Již do výběru práce jsem šel s tím, abych se naučil to, co by mě v budoucnu bavilo a čím bych se také později mohl živit. Naučil jsem se kompletně zanalyzovat projekt. V minulosti mi přišla analýza celkem zbytečná, ale zjistil jsem, že může být naopak velice přínosná a usnadní práci při programování. Naučil jsem se také orientovat a vyvíjet aplikace v minulosti mi absolutně neznámém Yii frameworku, který navíc využívá objektově orientované programování v PHP.

Samotný projekt má své nedostatky a jsem si toho plně vědom. Bohužel některé věci už jsem nestihl dodělat. Jsou zde věci, které jsem měl v plánu, ale na jejich realizaci mi nezbyl čas. Mezi ně patří například zařazení captcha kódu při registraci uživatele. Po registraci by mohl být uživateli zaslán ověřovací e-mail. Také by mohl mít uživatel v nastavení možnost zasílání pravidelných reportů se statistikami dostupnosti serverů.

V budoucnu bych se chtěl tomuto projektu dále věnovat, jelikož u nás mnoho podobných aplikací pro monitorování serverů není. Pokud bych vylepšil a doplnil funkce, které jsem popsal výš, tak si myslím, že by moje aplikace mohla být slušnou alternativou k těm stávajícím.

Literatura

[1] BRUCKNER, Tomáš a Miroslav MÜLLER. *Tvorba informačních systémů: principy, metodiky, architektury*. 1. vyd. Praha: Grada, 2012, 357 s. Management v informační společnosti. ISBN 978-80-247-4153-6.

[2] KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. 2. aktualiz. vyd. Brno: Computer Press, 2006, 176 s. ISBN 80-251-1083-4.

[3] BRÁZA, Jiří. *PHP 5: začínáme programovat*. 1. vyd. Praha: Grada Publishing, 2005, 244 s. ISBN 80-247-1146-X.

[4] DRUSKA, Peter. *CSS a XHTML: tvorba dokonalých webových stránek krok za krokem*. 1 vyd. Praha: Grada, 2006, 200 s. ISBN 80-247-1382-9.

[5] HOLZSCHLAG, Molly E. *HTML a CSS: jdi do toho*. 1. vyd. Překlad Jan Bouda. Praha: Grada, 2006, 263 s. ISBN 80-247-1454-X.

[6] *JQuery cookbook*. 1st ed. Cambridge: O'Reilly, c2010, 451 s. ISBN 05-961-5977-3.

[7] PROCHÁZKA, David. *PHP 6: začínáme programovat*. 1. vyd. Praha: Grada, 2012, 183 s. Průvodce (Grada). ISBN 978-80-247-3899-4.

[8] WINESETT, Jeffrey. *Agile web application development with Yii1.1 and PHP5: fast-track your web application development by harnessing the power of the Yii PHP framework*. Birmingham: Packt Publishing, 2010, 348 s. ISBN 978-1-847199-58-4.

[9] „Model-View-Controller (MVC),“ Yii framework, 1 Květen 2012. [Online]. Available: http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc. [Přístup získán 1. Květen 2012].

[10] „Model,“ Yii framework, 1. Květen 2012. [Online]. Available: http://www.yiiframework.com/doc/guide/1.1/en/basics.model. [Přístup získán 1. Květen 2012].

[11] „View,“ Yii framework, 1. Květen 2012. [Online]. Available: http://www.yiiframework.com/doc/guide/1.1/en/basics.view. [Přístup získán 1. Květen 2012].

[12] „Controller,“ Yii framework, 1. Květen 2012. [Online]. Available: http://www.yiiframework.com/doc/guide/1.1/en/basics.controller. [Přístup získán 1. Květen 2012].

[13] PROCHÁZKA, David. *První kroky s internetem*. 3., aktualiz. vyd. Praha: Grada, 2010. ISBN 80-247-3255-6.

[14] KRČMÁŘ, Petr. *Linux: postavte si počítačovou síť*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-1290-1.

[15] ROSENBROCK, Eric a Eric FILSON. *Linux, Apache, MySQL a PHP: instalace a konfigurace prostředí pro pokročilé webové aplikace*. 1 vyd. Překlad Karel Voráček. Praha: Grada, 2005, 344 s. ISBN 80-247-1260-1.

Seznam obrázků

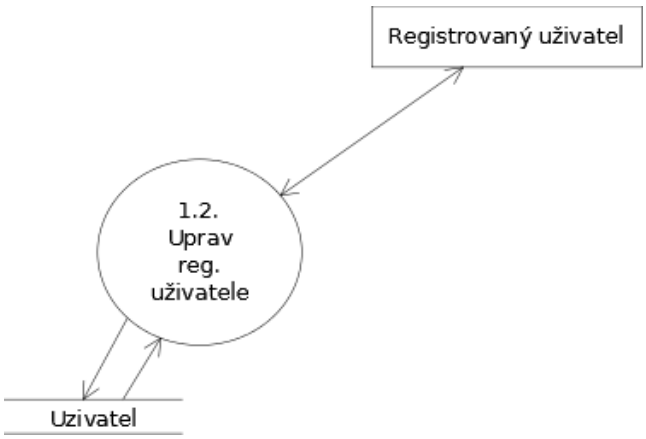
Přílohy

I. Příloha na CD/DVD

Přiložené DVD obsahuje následující soubory:
Aplikace – Hotová aplikace
BakalarskaPrace – Text bakalářské práce
Databaze – Databáze pro import
WebServer – Program potřebný pro běh aplikace
README-prvotni nastaveni aplikace.TXT – průvodce prvotním nastavením aplikace

II. Analýza

Minispecifikace funkce 1.2. Uprav registrovaného uživatele



Obrázek 20 : Minispecifikace 1.2. – Uprav registrovaného uživatele

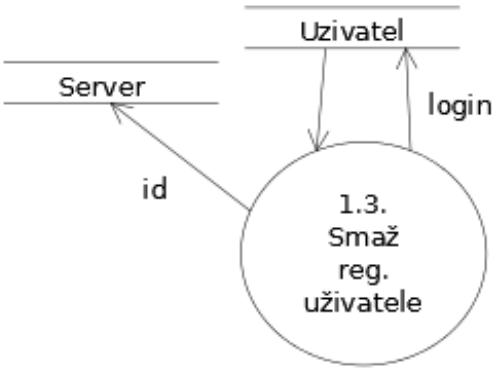
Algoritmus:

1. Zobraz formulář s údaji registrovaného uživatele z Uzivatel

Login:	Uživatel může upravit login, Uzivatel.login
Nové heslo:	Uživatel může upravit heslo
Původní heslo:	Uživatel zadá původní heslo, aby jej mohl změnit na nové
Jméno:	Uživatel může upravit jméno, Uzivatel.jmeno
Příjmení:	Uživatel může upravit příjmení, Uzivatel.prijmeni
Email:	Uživatel může upravit email, Uzivatel.email

2. Uživatel změní údaje pro úpravu svého profilu
3. Zkontroluj údaje zadané uživatelem, vyskytne-li se chyba, jdi zpět na krok 2. Je-li všechno v pořádku, pokračuj dále
4. Zapiš vyplněný formulář jako záznam do Uzivatel
5. Konec cyklu pro jednoho uživatele

Minispecifikace funkce 1.3. Smaž registrovaného uživatele



Obrázek 21 : Minispecifikace 1.3. – Smaž registrovaného uživatele

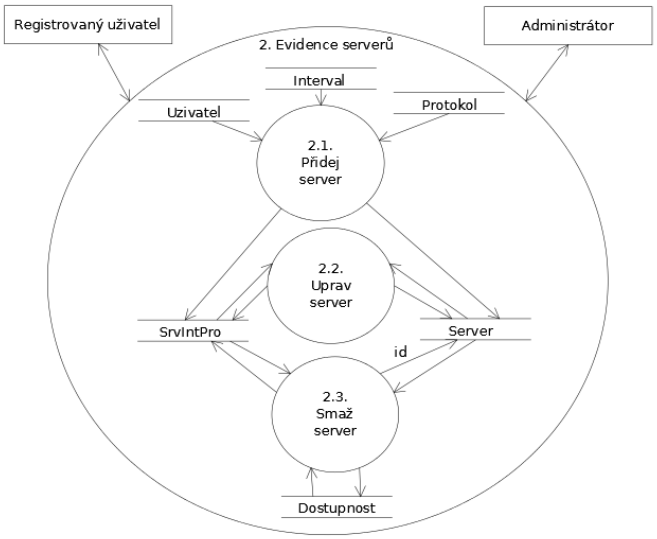
Algoritmus:

1. Zobraz seznam registrovaných uživatelů z Uzivatel

Login	Email	Jméno	Příjmení	Server	Protokol	Interval

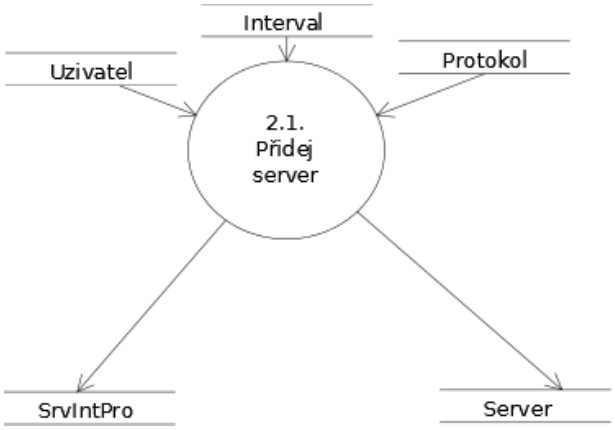
2. Administrátor vybere uživatele
3. Smaž záznam uživatele z Uzivatel
4. Automaticky smaž záznam z Server, ke kterému byl uživatel přiřazen
5. Konec cyklu pro jednoho uživatele

1.1.1.1. DFD 1. Úrovně – Evidence serverů



Obrázek 22 : DFD 1. Úrovně - Evidence serverů

Minispecifikace funkce 2.1. Přidej server



Obrázek 23 : Minispecifikace 2.1. – Přidej server

Algoritmus:

1. Zobraz formulář pro přidání nového serveru pro sledování

Url:	Uživatel zadá url
Název:	Uživatel může zadat název
Protokol:	Uživatel vybere ze seznamu protokol, data z Protokol
Interval:	Uživatel vybere ze seznamu interval, data z Interval
Stav:	Uživatel zadá stav, checkbox – měřit/neměřit

2. Uživatel vyplní formulář pro přidání serveru
3. Zkontroluj údaje zadané uživatelem, vyskytne-li se chyba, jdi zpět na krok 2. Je-li všechno v pořádku, pokračuj dále
4. Vygeneruj hodnotu číslo(id) pro atribut Server.id
5. Přiřaď hodnotu přihlášeného uživatele z Uzivatel.login do Server.Uzivatel_login
6. Zapiš data do Server a SrvIntPro
7. Konec cyklu pro jeden server

Minispecifikace funkce 2.2. Uprav server

Obrázek 24 : Minispecifikace 2.2. – Uprav server

Algoritmus:

1. Zobraz seznam přidaných serverů z Server, SrvIntPro, Protokol, Interval

Url	Nazev	Protokol	Interval	Stav

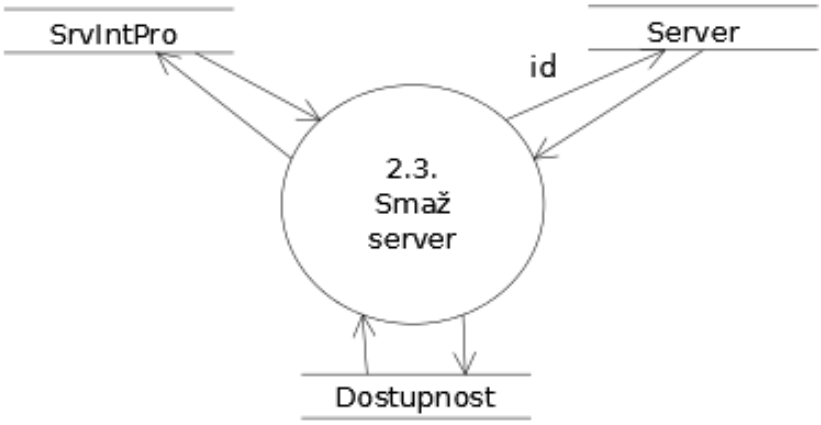
2. Registrovaný uživatel nebo Administrátor vybere server

3. Zobraz formulář pro úpravu serveru

Url:	Uživatel může upravit url, Server.url
Název:	Uživatel může upravit název, Server.nazev
Protokol:	Uživatel může upravit protokol, výběr ze seznamů protokolů, data z Protokol, Protokol.nazev
Interval:	Uživatel může upravit interval, výběr ze seznamů intervalů, data z Interval, Interval.nazev
Stav:	Uživatel může upravit stav, checkbox – měřit/neměřit, SrvIntPro.stav

4. Registrovaný uživatel nebo administrátor upraví požadované hodnoty
5. Zkontroluj údaje zadané registrovaným uživatelem nebo administrátorem, vyskytne-li se chyba, jdi zpět na krok 4. Je-li všechno v pořádku, pokračuj dále
6. Zapiš vyplněný formulář jako záznam do Server a do SrvIntPro
7. Konec cyklu pro úpravu jednoho serveru

Minispecifikace funkce 2.3. Smaž server



Obrázek 25 : Minispecifikace 2.3. – Smaž server

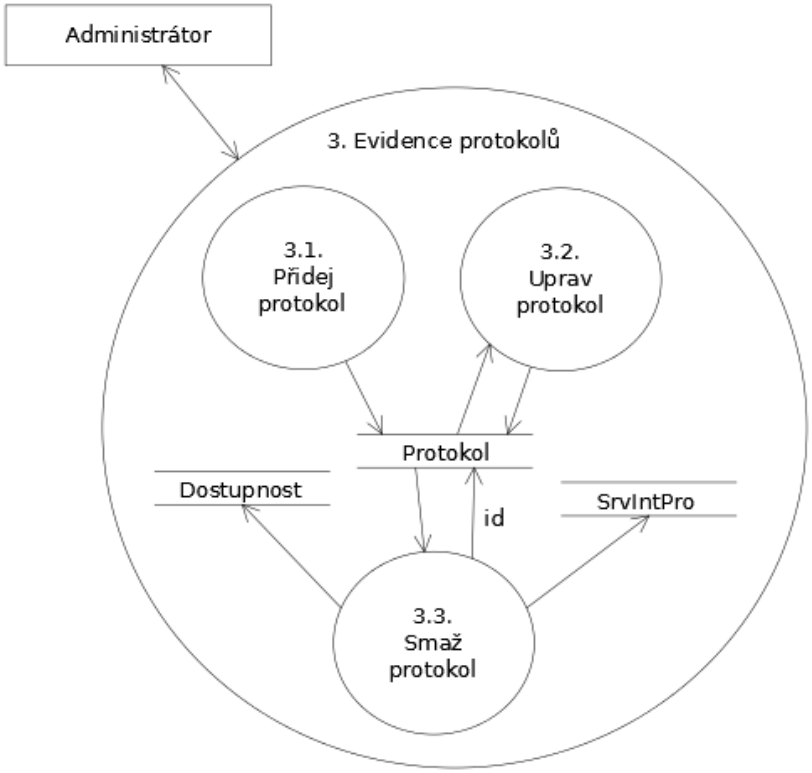
Algoritmus:

1. Zobraz seznam serverů z Server, SrvIntPro, Protokol, Interval

Url	Nazev	Protokol	Interval	Stav

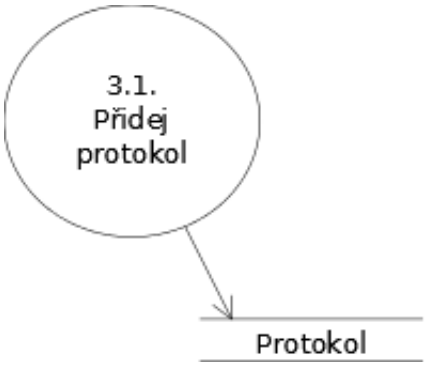
2. Registrovaný uživatel nebo administrátor vybere server
3. Smaž záznam serveru z Server
4. Automaticky smaž záznam z SrvIntPro, ke kterému byl server přiřazen
5. Automaticky smaž záznam z Dostupnost, ke kterému byl server přiřazen
6. Konec cyklu pro smazání jednoho serveru

1.1.1.2. DFD 1. Úrovně – Evidence protokolů



Obrázek 26 : DFD 1. Úrovně - Evidence protokolů

Minispecifikace funkce 3.1. Přidej protokol



Obrázek 27 : Minispecifikace 3.1. – Přidej protokol

Algoritmus:

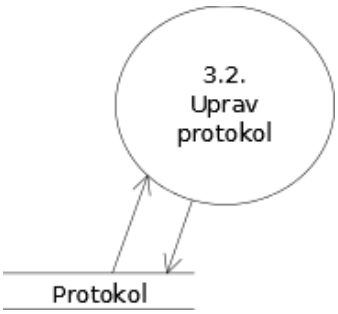
1. Zobraz formulář pro přidání nového protokolu

Protokol:	Administrátor zadá protokol

2. Administrátor vyplní formulář pro přidání protokolu

3. Zkontroluj údaje zadané administrátorem, vyskytne-li se chyba, jdi zpět na krok 2. Je-li všechno v pořádku, pokračuj dále
4. Vygeneruj hodnotu číslo(id) pro atribut Protokol.id
5. Zapiš data do Protokol
6. Konec cyklu pro přidání jednoho protokolu

Minispecifikace funkce 3.2. Uprav protokol



Obrázek 28 : Minispecifikace 3.2. – Uprav protokol

Algoritmus:

1. Zobraz seznam protokolů z Protokol

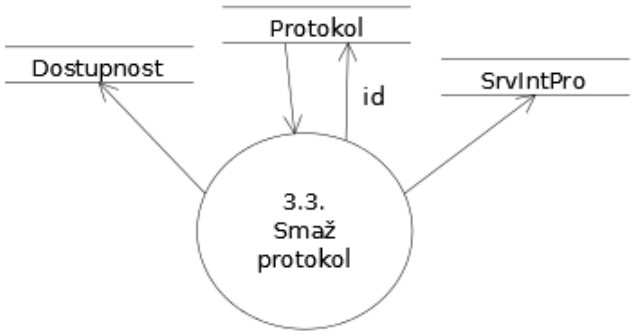
Protokol

2. Administrátor vybere Protokol
3. Zobraz formulář pro úpravu Protokolu

Protokol:	Administrátor může upravit protokol, data z Protokol, Protokol.protokol

4. Administrátor upraví záznam protokolu z Protokol
5. Zkontroluj údaje zadané administrátorem, vyskytne-li se chyba, jdi zpět na krok 4. Je-li všechno v pořádku, pokračuj dále
6. Zapiš vyplněný formulář jako záznam do Protokol
7. Konec cyklu pro úpravu jednoho protokolu

Minispecifikace funkce 3.3. Smaž protokol



Obrázek 29 : Minispecifikace 3.3. – Smaž protokol

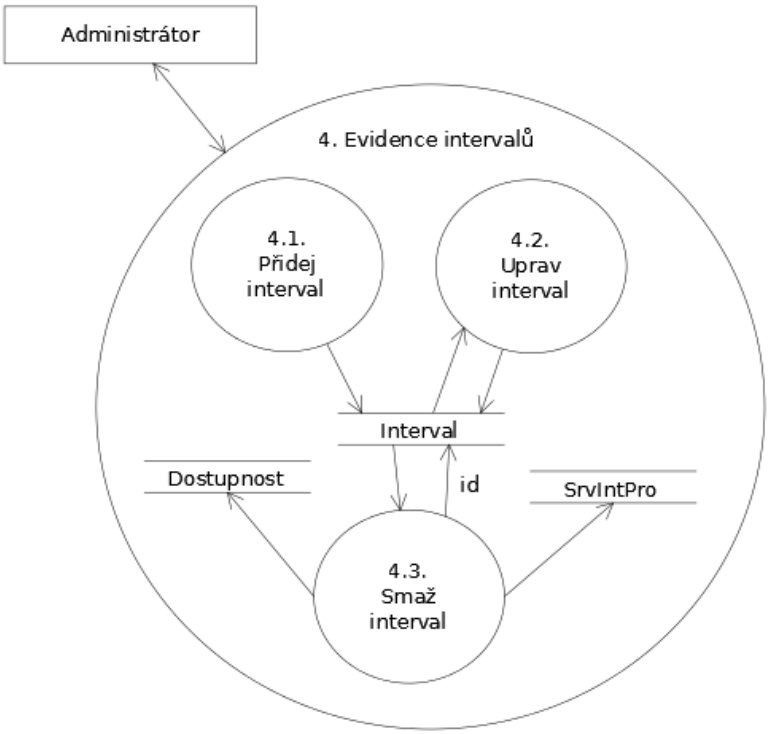
Algoritmus:

1. Zobraz seznam protokolů z Protokol

Protokol

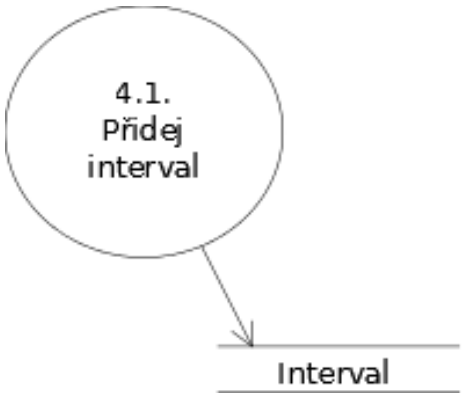
2. Administrátor vybere Protokol
3. Smaž záznam protokolu z Protokol
4. Automaticky smaž záznamy z SrvIntPro, ke kterým byl protokol přiřazen
5. Automaticky smaž záznamy z Dostupnost, ke kterým byl protokol přiřazen
6. Konec cyklu pro smazání jednoho protokolu

1.1.1.3. DFD 1. Úrovně – Evidence intervalů



Obrázek 30 : DFD 1. Úrovně - Evidence intervalů

Minispecifikace funkce 4.1. Přidej interval



Obrázek 31 : Minispecifikace 4.1. – Přidej interval

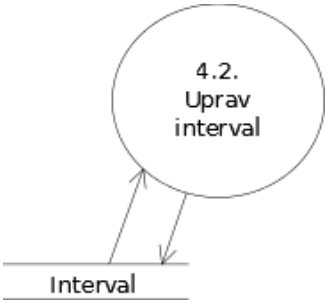
Algoritmus:

1. Zobraz formulář pro přidání nového intervalu

Interval:	Administrátor zadá Interval

2. Administrátor vyplní formulář pro přidání intervalu
3. Zkontroluj údaje zadané administrátorem, vyskytne-li se chyba, jdi zpět na krok 2. Je-li všechno v pořádku, pokračuj dále
4. Vygeneruj hodnotu číslo(id) pro atribut Interval.id
5. Zapiš data do Interval
6. Konec cyklu pro přidání jednoho Intervalu

Minispecifikace funkce 3.2. Uprav Interval



Obrázek 32 : Minispecifikace 4.2. – Uprav interval

Algoritmus:

1. Zobraz seznam intervalů z interval

Interval

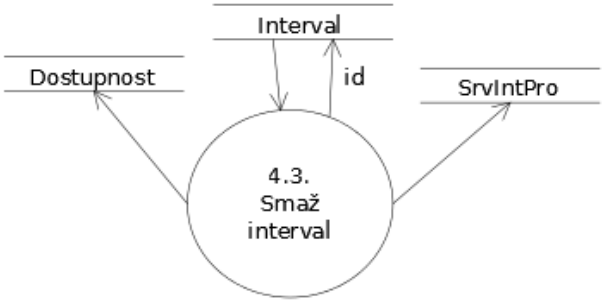
2. Administrátor vybere interval
3. Zobraz formulář pro úpravu interavu

Interval:	Administrátor může upravit interval, data z Interval, Interval.interval

4. Administrátor upraví záznam intervalu z Interval

5. Zkontroluj údaje zadané administrátorem, vyskytne-li se chyba, jdi zpět na krok 4. Je-li všechno v pořádku, pokračuj dále
6. Zapiš vyplněný formulář jako záznam do Interval
7. Konec cyklu pro úpravu jednoho intervalu

Minispecifikace funkce 4.3. Smaž interval



Obrázek 33 : Minispecifikace 4.3. – Smaž interval

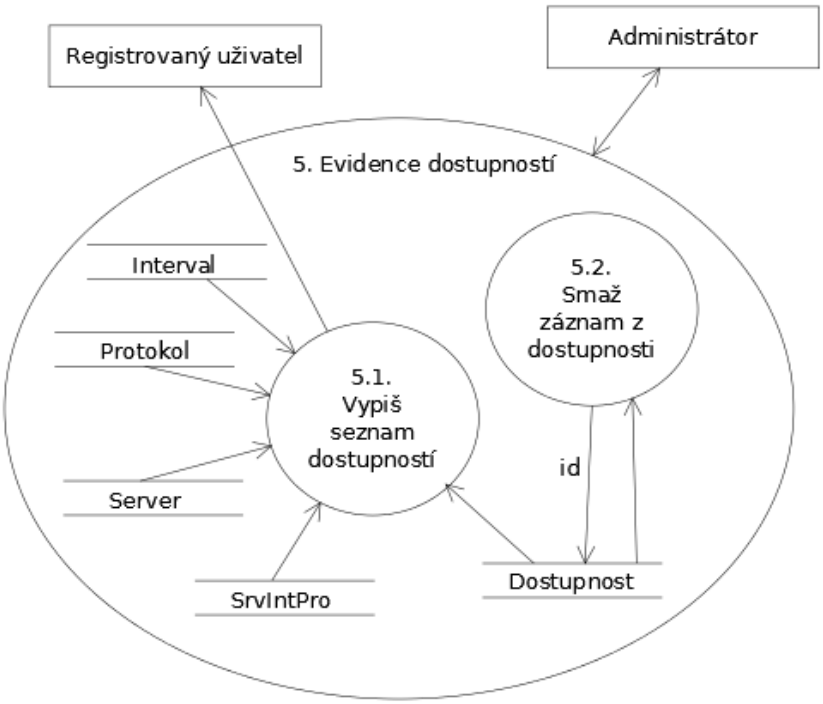
Algoritmus:

1. Zobraz seznam intervalů z inetrval

Protokol

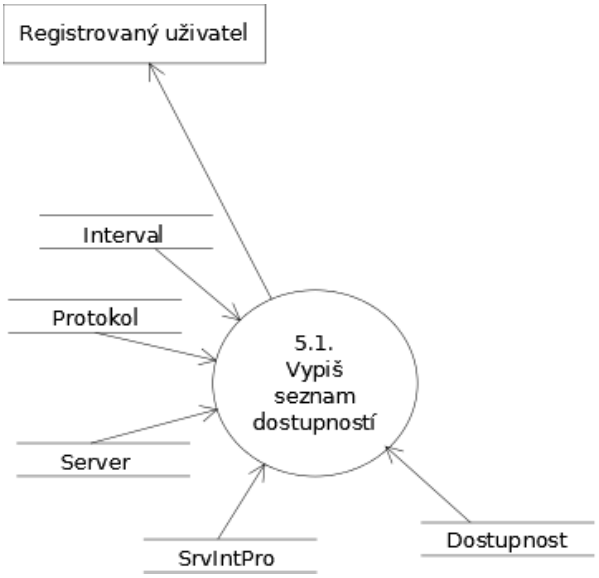
2. Administrátor vybere Interval
3. Smaž záznam intervalu z Interval
4. Automaticky smaž záznamy z SrvIntPro, ke kterým byl interval přiřazen
5. Automaticky smaž záznamy z Dostupnost, ke kterým byl interval přiřazen
6. Konec cyklu pro smazání jednoho intervalu

1.1.1.4. DFD 1. Úrovně – Evidence dostupností



Obrázek 34 : DFD 1. Úrovně - Evidence dostupností

Minispecifikace funkce 5.1. Vypiš seznam dostupností



Obrázek 35 : Minispecifikace 5.1. – Vypiš seznam dostupností

Algoritmus:

1. Zobraz seznam dostupností z Dostupnost, Interval, Protokol, Server, SrvIntPro

Datum a čas	url	Název	Návratový kód	Protokol	Interval	Odezva

2. Konec cyklu pro zobrazení seznamu dostupností

Minispecifikace funkce 5.2. Smaž záznam z dostupnosti

Obrázek 36 : Minispecifikace 5.2. – Smaž záznam z dostupnosti

Algoritmus:

1. Zobraz seznam dostupností z Dostupnost

Datum a čas	url	Název	Návratový kód	Protokol	Interval	Odezva

2. Administrátor vybere dostupnost
3. Smaž záznam dostupnosti z Dostupnost
4. Konec cyklu pro smazání jedné dostupnosti